the APL perspective

(formerly, the *SWAPL Newsletter*)

August, 1993

To APL93 Attendees:

Welcome to APL93. We hope that you enjoy the week and will take home fond memories of a great meeting.

This newsletter is the official publication of the SouthWest APL Users' Group (SWAPL), a user group based in Dallas, Texas. SWAPL has been meeting for over twenty years. This newsletter is a monthly publication, generally from sixteen to twenty-four pages in length, and has been published continuously for the past fourteen years. Originally, our readership was predominantly in the southwestern part of the United States. Our coverage has expanded and today most of our readers come from outside the southwestern United States, including quite a few with European addresses, and some in Russia and the Far East.

Our newsletter contains notes from meetings of user groups around the US (whichever are willing to contribute -- SWAPL meetings always appear) and future programs for upcoming meetings; letters to the Editor; APL product announcements; occasional advertising and job announcements; and articles on APL topics. There is also a listing of local user groups and a contact point for each. Since October, 1987, there has been a monthly column on second generation APL topics. (The column is currently called "⍝ *EXTENSION NOTES*".) More articles are being submitted on a regular basis by other authors.

We hope that you read and enjoy your complimentary issue of our publication, *the APL perspective*.

-- Stuart Yarus

"A rose by any other name..."

The *SWAPL Newsletter* has changed its name! The SouthWest APL Users' Group has been discussing this name change for several months, and now it has finally happened. Over 120 proposed names were submitted. The new name suggested by Todd Marshall, *the APL perspective*, reflects the national and international readership, authorship, and coverage of the monthly newsletter.

The intent of *the APL perspective* is to provide its readership with: current news in the APL world, including new product announcements and reviews; articles on all aspects of all dialects of APL, including J; techniques and algorithms; operating system interfaces; graphical user interfaces; and last but not least, news in the non-APL world which may affect the APL world.

Table of Contents

Welcome to APL93	1
Newsletter name change	1
Letters to the Editor	2
SIGAPL Election Results	3
Working with Manugistics' APLWIN	4
Mental Flexibility Test	9
How to Become an APLSysop, & Why	11
Tidbits	12
User Group News	12
Dallas July Meeting Notes	13
APL User Groups (USA)	13
APL2 for OS/2	14
APL*PLUS II GUI Reference Card	16
Guidelines for Graphical User Interfaces	22
Application Notes	23
REXX/XEDIT Windowing Interface to APL2	26
The Soundex Method	30
⍝ Extension Notes	32

the APL perspective

published by the SouthWest APL Users' Group,
P. O. Box 210367, Bedford, Texas 76095, USA
e-mail: 73700.2545@CompuServe.com

Editor Stuart Yarus
73700.2545@CompuServe.com

Editorial Staff Margaret Brooker
72124.271@CompuServe.com
Rick Butterworth
71420.742@CompuServe.com
Todd Marshall
70544.715@CompuServe.com
Gregg Taylor
71033.372@CompuServe.com

Articles

The APL Perspective staff encourages you (and your grandmother from the old country, as long as she still speaks APL) to submit articles for publication. The preferred media is floppy diskette (PC format, 5¼ inch or 3½ inch size). E-mail is quite acceptable. Sneakernet at SWAPL meetings also works very well. *The APL Perspective* is composed on an Intel 486-based personal computer using Ami Professional under Windows 3.1. Articles formatted in Word Perfect, Word for Windows, and WordStar have been easily imported into Ami Pro, as have flat (unformatted) files and data in the Rich Text Format. Printing is done on a Hewlett-Packard LaserJet III.

Advertising

The APL Perspective does take advertising. Our rates are $40.00 for a full page, and are scaled down for multiple insertions and smaller ads. Please contact our acting Advertising Manager (the Editor) for details.

Back Copies

Back copies of issues are available from July, 1991, and cost US$2.00 each. There is an extra postage charge for non-US addresses.

Copyright

Copyright 1993 by *the APL perspective*. Articles without specific copyright notices may be reproduced, unchanged, by other not-for-profit User Groups if credit is given to author and publication.

Letters to the Editor

Dear Editor:

I feel compelled to make some clarifications regarding the review of my presentation at the June meeting of the SouthWest APL Users' Group. In the limited time that I had, and with the large amount of material that could have been covered, I think that I neglected to make clear several ideas that I expounded during that meeting. In the interest of accuracy, I would like to make the following points:

a) Whether the APL community likes it or not, APL is in a "catch-up" mode in the Windows world. Why else do we emulate (in name and style) the Visual Basic (VB) paradigm? Working (as I get paid to do) in VB, I am very aware of the differences between it and Dyadic's Dyalog APL/W. There are differences; the main one being the working environment. Dyalog APL/W Release 6.3 is functionally a lot closer to the VB paradigm. And, while APL would never want to reproduce the functionality and environment *in toto*, comparisons will continue to be made *vis-a-vis* VB.

b) Working in the Windows environment is possible by virtue of the programmer's ability to access the Windows DLLs (Dynamic Link Libraries). DLLs are libraries of function calls; these function calls support all of the Windows functionality. Just because Dyadic chose to implement a suite of quad functions to ease the interface task in no way implies that the Windows functionality *not* implemented in these calls is therefore not available. In fact, all the functionality *is* provided by accessing the DLLs through the `⎕NA` system function. After mapping the C-style arguments to APL-style arguments, the Windows calls may be executed as though they were APL functions. (This point I completely failed to make clear at the meeting.)

c) When I made the statement that APL will always be about a year behind VB, I was referring to the "controls" that may be placed on the forms. Controls are things like scrollbars, input boxes, listboxes, labels, etc. -- objects that the user interacts with when working with his data. These controls can be anything you want them to be (or that someone programmed them to be). For instance for $250, you can get a (VB) control that is a quite sophisticated spreadsheet -- you just provide the control a data array, it does the rest. The catch here is that this control was written specifically for VB -- C programs can't even tap into its functionality! If APL is to keep up with these capabilities, either someone is going to have to figure out how to integrate this class of

objects (that have a .VBX extension) into APL, or someone will have to re-write the functionality for us. Lack of access to these controls from the VB environment will continue to keep APL "behind" the leader. (As I have written in an earlier edition of this newsletter, the amount of money invested in adding controls and support for the VB environment cannot be matched by the APL community.)

d) Indeed, I think it is true that the Dyalog APL/W architecture is not "open". I am not sure that it is possible, at this time, for any individual (or company) to add to the toolbox of controls, and their properties, available in APL/W. Perhaps the Auxiliary Processor capabilities allow an opportunity for this, but I am not clear on how these are related. I think it would be nice to be able to add controls to Dyalog's arsenal, and perhaps Dyadic Systems could either provide documentation for how this could be done, or an explanation of why it's not possible.

The Windows environment offers a huge, new, and different resource to tap. It is going to take time for all of us to get used to its concepts, its impact, and how we are to relate to it. It is by having these presentations and discussions that we come to understand it better.

Gregg Taylor
Keller, Texas

Dear Editor:

Thanks for July issue of the newsletter and for printing my note on Gregg's function. AS USUAL, I had a good time reading this WELL DONE journal. There is a little mistake there, on page 23. The output of `DISPLAY ⊂ 'TOM' 'DICK' 'AND' 'MARY'` (line 2 of APL inputs) should have one more line around it. It's a scalar, not a vector.

Alexander Skoromokhov
Obninsk, Russia

[Curtis Jones (San Jose, California) also noted the error. We regret the misprint. -Ed]

SIGAPL Election Results

Election results for SIGAPL, the Special Interest Group for APL of the Association for Computing Machinery (ACM), have recently been announced. The elected positions and new officers are: Chair, Dick Bowman; Vice Chair, Stuart Yarus; Secretary-Treasurer, Mike Kent; Members-at-Large (in alphabetical order): Dick Holt, Chris Lee, David Weintraub. The officers were elected for the term July 1, 1993, through June 30, 1995. In order to better serve SIGAPL members and the APL community at large, ACM has established mail forwarding accounts for the elected officers and newsletter editors. These accounts have also been established for all other ACM SIGs. The SIGAPL forwarding accounts, and the addresses for the Members-at-Large are:

Dick Bowman	CHAIR_SIGAPL@ACM.org
Stuart Yarus	VC_SIGAPL@ACM.org
Mike Kent	SECRETARY_SIGAPL@ACM.org
Mike Kent	TREASURER_SIGAPL@ACM.org
Quote Quad	EDITORS_SIGAPL@ACM.org
Dick Holt	dick.holt@ACM.org
Chris Lee	71542.606@CompuServe.com
David Weintraub	dave_weintraub@jhuapl.edu
Lynne Shaw	Lynne_Shaw@mts.cc.Wayne.edu.

Lynne Shaw is immediate Past Chair of SIGAPL.

Everyone, whether or not a member of SIGAPL, is encouraged to submit e-mail to any SIGAPL officer. E-mail to the Internet can be sent from CompuServe, MCI Mail, Bix, Delphi, and America On-Line.

The current SIGAPL appointed officer positions have not yet been filled. They are: Recording Secretary, Membership Coordinator, Conference Coordinator, and Quote Quad Editor(s). All SIGAPL officers are volunteers.

SIGAPL is an international organization dedicated to "promoting the development and application of the APL programming language". The success of SIGAPL will be due largely to the efforts of people in the world-wide APL community who choose to become involved with SIGAPL. Please choose to get involved. There is much to be done, from very small to very large tasks.

Your SIGAPL officers.

WORKING WITH MANUGISTICS' APLWIN

Todd B. Marshall
Conroe, Texas
e-mail: 70544.715@CompuServe.com

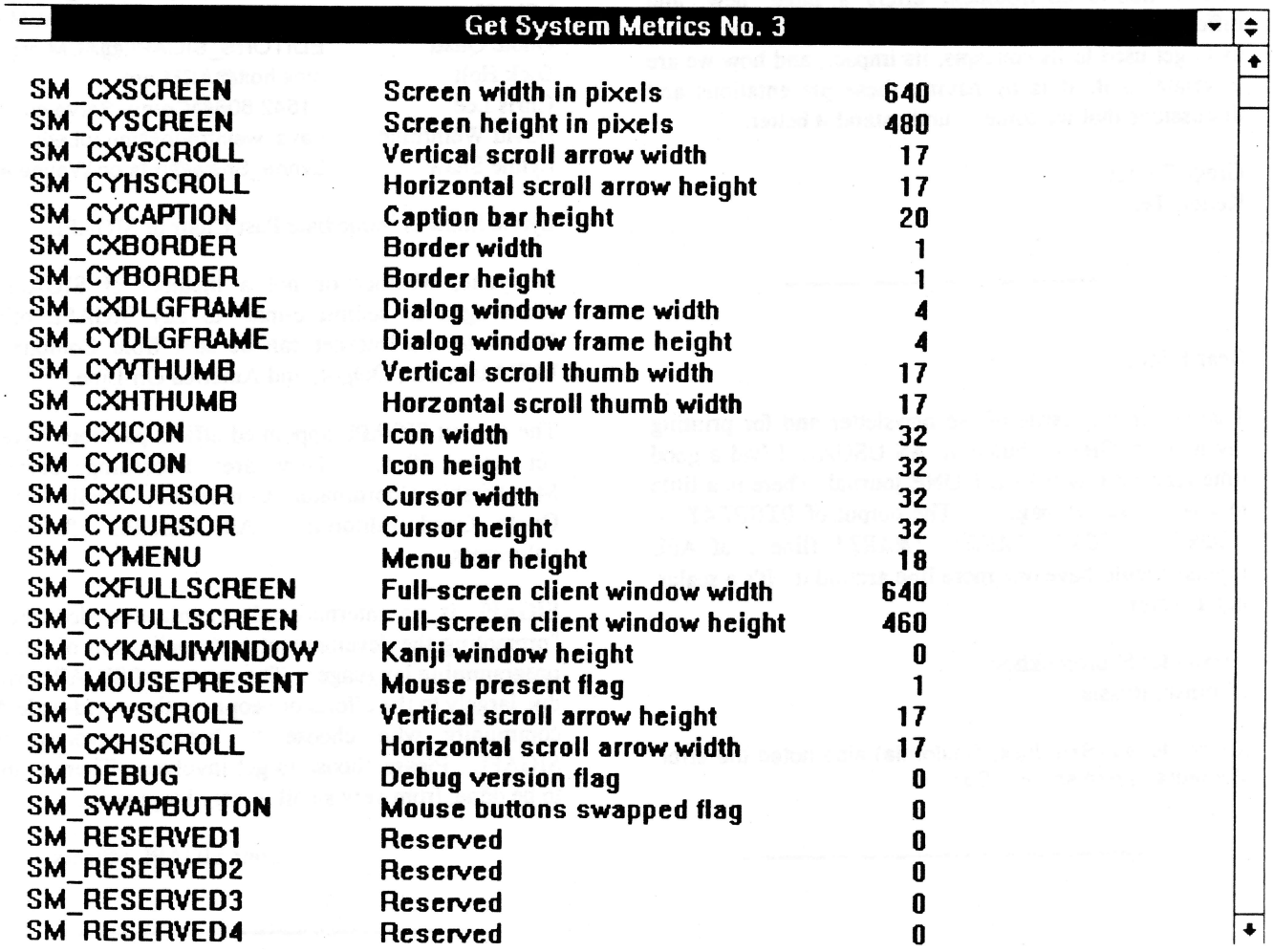
In my first (two-part) article I laid the groundwork for interfacing APL*PLUS to the Windows API through the facilities of APLWIN. It was painful but the worst is pretty much behind us. This article assumes an understanding of the material presented before. We continue to follow along in Charles Petzold's Programming Windows.

Once Charles has introduced us to the mechanism of making noise on the screen, he leads us through some concepts of controlling that noise. To introduce the concepts, his examples SYSMETS1 through SYSMETS3 display the "system metrics" values in windows. He also introduces the concept of horizontal and vertical scrolling of a window. Knowledge of these system metrics is necessary for repainting windows in response to size changes and scrolling.

In my function SYSMETS3, my main program is implemented in lines 1 through 10. As in my previous articles, my function is recursive and is invoked with the command:

```
SYSMETS3 0
```

As Windows sends messages which are dispatched by our filter, the function is called again recursively with a non-zero argument. The argument is the branch target for line 4.


Get System Metrics No. 3
SM_CXSCREEN Screen width in pixels 640
SM_CYSCREEN Screen height in pixels 480
SM_CXVSCROLL Vertical scroll arrow width 17
SM_CYHSCROLL Horizontal scroll arrow height 17
SM_CYCAPTION Caption bar height 20
SM_CXBORDER Border width 1
SM_CYBORDER Border height 1
SM_CXDLGFRAME Dialog window frame width 4
SM_CYDLGFRAME Dialog window frame height 4
SM_CYVTHUMB Vertical scroll thumb width 17
SM_CXHTHUMB Horzontal scroll thumb width 17
SM_CXICON Icon width 32
SM_CYICON Icon height 32
SM_CXCURSOR Cursor width 32
SM_CYCURSOR Cursor height 32
SM_CYMENU Menu bar height 18
SM_CXFULLSCREEN Full-screen client window width 640
SM_CYFULLSCREEN Full-screen client window height 460
SM_CYKANJIWINDOW Kanji window height 0
SM_MOUSEPRESENT Mouse present flag 1
SM_CYVSCROLL Vertical scroll arrow height 17
SM_CXHSCROLL Horizontal scroll arrow width 17
SM_DEBUG Debug version flag 0
SM_SWAPBUTTON Mouse buttons swapped flag 0
SM_RESERVED1 Reserved 0
SM_RESERVED2 Reserved 0
SM_RESERVED3 Reserved 0
SM_RESERVED4 Reserved 0

On lines 6, 7, and 8 I use an in-line subroutine technique. The APL functions SUBR and RETURN work together to provide this programming style. They allow me to modularize my code "within" an APL function. I get the advantage of well structured code without the annoyance of many small functions.

I have arranged the SYSMETS3 program into three principal sections. They are the Main Control Section: lines 1 through 12; the Message Service Section: lines 12 through 104; and the In-Line Subroutines Section: lines 105 through 193. All program exits are made through line 195. Within each of these sections, I maintain the subsections in alphabetical order so I can quickly find the code of interest.

MESSAGE SUBCLASSING:

In this example I introduce you to message subclassing as supported by APLWIN. Lines 128 through 152 define our message filter. This filter handles scrolling messages (WM_VSCROLL and WM_HSCROLL) which contain submessages (e.g. SB_TOP, SB_BOTTOM, etc.). The W_CreateFilter function accepts a hierarchical array of filter descriptions. As you can see for WM_VSCROLL, a form of structured programming using the "BEGIN" and "END" keywords allows us to define our message subclasses. Reading the code, you can see that it becomes quite straight forward to deal with user clicking on the various objects which make up a scroll bar.

Next month we will follow along with Petzold and deal with the keyboard through Windows.

```
    ∇ SYSMETS3 MSG;⎕IO
[1]   ⍝SYSMET3 -- System Metrics Display Program No. 3
[2]   ⎕IO←1
[3]   ⍝Branch to Service Routine if MSG ≠ 0
[4]   →(0≠MSG)/MSG
[5]   ⍝Form the sysmetrics table
[6]   →SUBR MAKE_SYSMETS_TBL⍝ Define the System Metrics Table
[7]   →SUBR DEF_MSG_FILTER  ⍝ Define the Windows Message Filter
[8]   →SUBR CREATE_WINDOW   ⍝ Create the Window
[9]   rc←W_Wait             ⍝ Wait for things to happen
[10]  →Exit                 ⍝ Exit when window is closed
[11]
[12]  ⍝=============================================================================
[13]  ⍝Message Service Routines
[14]  ⍝-----------------------------------------------------------
[15]  WM_CREATE:
[16]  hwnd←1⊃w_args ⋄ hdc←W_Call 'GetDC' hwnd ⍝ Get device context
[17]  (rc tm)←W_Call 'GetTextMetrics' hdc ''
[18]  tm←⎕ENLIST (C2I 16↑tm) ((⎕AV⍳ 9↑16↓tm)-⎕IO) (C2I ¯6↑tm)
[19]  cxChar←tm[6]     ⍝tm.tmAveCharWidth
[20]  cxCaps←⌊cxChar×(1 1.5)[⎕io+2|tm[16]]  ⍝ (tmPitchAndFamily & 1?3:2) * cxChar/2
[21]  cyChar←+/tm[1 5]         ⍝tm.tmHeight
[22]  rc←W_Call 'ReleaseDC' hwnd hdc
[23]  nMaxWidth←40 18 +.× cxChar,cxCaps
[24]  ⍝Get all the system metrics
[25]  metx←⊂[2](⊂'GetSystemMetrics'),[1.5]1⊃¨sysmetrics
[26]  dmetx←W_CallEach metx
[27]  NUMLINES←0⍴⍴sysmetrics
[28]  ⍝Variables C Assumes to be initialized to 0 at start
[29]    nVscrollInc ← nHscrolling ← nVscrollPos ← nVscrollPos ← nHscrollMax ←
nHscrollPos ← 0
[30]  →Exit
[31]
[32]  ⍝-----------------------------------------------------------
[33]  WM_DESTROY:⍝ Terminate the Application
[34]  rc←W_Call'PostQuitMessage' 0
```

```
[35]  →Exit
[36]
[37]  ⍝-------------------------------------------------------
[38]  ⍝ WM_HSCROLL:⍝Deal with the Scrolling Messages
[39]  HSB_LINEUP: nHScrollInc←-1 ◊ →WM_HSBREAK
[40]  HSB_LINEDOWN:nHscrollInc←1 ◊ →WM_HSBREAK
[41]  HSB_PAGEUP:nHscrollInc←-8 ◊ →WM_HSBREAK
[42]  HSB_PAGEDOWN:nHscrollInc←8 ◊ →WM_HSBREAK
[43]  HSB_THUMBPOSITION:nHscrollInc←(3⊃w_args)-nHscrollPos ◊ →WM_HSBREAK
[44]  WM_HSBREAK:
[45]  →(0=nHscrollInc←((-nHscrollPos)⌈nHscrollInc⌊nHscrollMax-nHscrollPos))/Exit
[46]  nHscrollPos←nHscrollPos+nHscrollInc
[47]  W_B 'ScrollWindow' hwnd (-cxChar×nHscrollInc) 0 0 0
[48]  W_B 'SetScrollPos' hwnd 'SB_HORZ' nHscrollPos 1
[49]  t←W_CallEachDebug W_B⍬
[50]  →Exit
[51]
[52]  ⍝-------------------------------------------------------
[53]  WM_PAINT:⍝ Paint the Screen
[54]  (hdc ps)←W_Call 'BeginPaint' w_args ''◊rect←C2I ps[4+⍳8]
[55]  nPaintBeg ← 0⌈nVscrollPos+¯1+rect[2]÷cyChar
[56]  nPaintEnd ← NUMLINES⌊nVscrollPos+rect[4]÷cyChar
[57]  i←⌊nPaintBeg◊t←W_B⍬
[58]  WM_LPPAINT:→(nPaintEnd<i←i+1)/WM_DONEPAINT
[59]  x←cxChar×1-nHscrollPos
[60]  y←cyChar×i-nVscrollPos
[61]  W_B'TextOut' hdc x y 1 (⍴⍴1←⊃i 1⊃sysmetrics)
[62]  W_B'TextOut' hdc (x+18×cxCaps) y 1 (⍴⍴1←i 2⊃sysmetrics)
[63]  W_B'SetTextAlign' hdc 'TA_RIGHT TA_TOP'
[64]  W_B'TextOut' hdc (x+40 18+.×cxChar,cxCaps) y 1 (⍴⍴1←5 0⍕i⊃dmetx)
[65]  W_B'SetTextAlign' hdc 'TA_LEFT TA_TOP'
[66]  →WM_LPPAINT
[67]  WM_DONEPAINT:
[68]  W_B 'EndPaint' w_args ps
[69]  t←W_CallEachDebug W_B⍬
[70]  →Exit
[71]
[72]  ⍝-------------------------------------------------------------
[73]  WM_SIZE:⍝Determine Screen Size
[74]  (cxClient cyClient)←w_args[3 4]
[75]  nVscrollMax←0⌈NUMLINES+2-cyClient÷cyChar
[76]  nVscrollPos←nVscrollPos⌊nVscrollMax
[77]  W_B 'SetScrollRange' hwnd 'SB_VERT' 0 nVscrollMax 0
[78]  W_B 'SetScrollPos' hwnd 'SB_VERT' nVscrollPos 1
[79]  t←W_CallEachDebug W_B⍬
[80]  nHscrollMax←0⌈2+(nMaxWidth-cxClient)÷cxChar
[81]  nHscrollPos←nHscrollPos⌊nHscrollMax
[82]  W_B 'SetScrollRange' hwnd 'SB_HORZ' 0 nHscrollMax 0
[83]  W_B 'SetScrollPos' hwnd 'SB_HORZ' nHscrollPos 1
[84]  t←W_CallEachDebug W_B⍬
[85]  →Exit
[86]
[87]  ⍝-------------------------------------------------------
[88]  ⍝ WM_VSCROLL:⍝Deal with the Scrolling Messages
[89]  VSB_TOP: nVscrollInc←-nVscrollPos◊→WM_VSBREAK
[90]  VSB_BOTTOM: nVscrollInc←nVscrollMax-nVscrollPos ◊ →WM_VSBREAK
```

```
[91]  VSB_LINEUP: nVscrollInc←-1 ◊ →WM_VSBREAK
[92]  VSB_LINEDOWN:nVscrollInc←1 ◊ →WM_VSBREAK
[93]  VSB_PAGEUP:nVscrollInc←¯1⌊-cyClient÷cyChar ◊ →WM_VSBREAK
[94]  VSB_PAGEDOWN:nVscrollInc←1⌈cyClient÷cyChar ◊ →WM_VSBREAK
[95]  VSB_THUMBTRACK:nVscrollInc←(3⊃w_args)-nVscrollPos◊ →WM_VSBREAK
[96]  WM_VSBREAK:
[97]  →(0=nVscrollInc←((-nVscrollPos)⌈nVscrollInc⌊nVscrollMax-nVscrollPos))/Exit
[98]  nVscrollPos←nVscrollPos+nVscrollInc
[99]  W_B 'ScrollWindow' hwnd 0 (-cyChar×nVscrollInc) 0 0
[100] W_B 'SetScrollPos' hwnd 'SB_VERT' nVscrollPos 1
[101] t←W_CallEachDebug W_B⍬
[102] →Exit
[103]
[104]⍝=============================================================
[105] ⍝ SUBROUTINES SECTION
[106] ⍝-------------------------------------------------------------
[107] CREATE_WINDOW:⍝ SUBROUTINE: Create the window
[108] ⍝Initialize this window
[109] (hInstance hPrevInstance lpszCmdParam nCmdShow)←W_Init
[110] szAppName←'SYSMETS3',⍕?1000
[111] ⍝Register this window class
[112] W_A ('CS_HREDRAW CS_VREDRAW') fp 0 0 hInstance
[113] W_A ('Load_IDI' 'IDI_APPLICATION') ('Load_IDC' 'IDC_ARROW')
[114] W_A ('GetStockObject' 'WHITE_BRUSH') 0 szAppName
[115] W_RegisterClass ⍕W_A⍬
[116]
[117] ⍝Create the Window
[118] W_A 'CreateWindow' szAppName 'Get System Metrics No. 3'
[119] W_A 'WS_OVERLAPPEDWINDOW WS_VSCROLL WS_HSCROLL'
[120] W_A 4⍴⊂'CW_USEDEFAULT'
[121] hwnd←W_Call ⍕W_A 0 0 hInstance 0
[122]
[123] ⍝Show the Window and Paint it
[124] t←W_CallEach ('ShowWindow' hwnd nCmdShow) ('UpdateWindow' hwnd)
[125] →RETURN
[126]
[127] ⍝-------------------------------------------------------------
[128] DEF_MSG_FILTER:⍝ SUBROUTINE: Define Message Filter
[129] W_FP 'WP'
[130] 'WM_PAINT'          W_FP THISFN WM_PAINT
[131] 'WM_DESTROY'        W_FP THISFN WM_DESTROY
[132] 'WM_CREATE'         W_FP THISFN WM_CREATE
[133] 'WM_SIZE'           W_FP THISFN WM_SIZE
[134] 'WM_VSCROLL'        W_FP 'BEGIN'
[135]   'SB_TOP'          W_FP THISFN VSB_TOP
[136]   'SB_BOTTOM'       W_FP THISFN VSB_BOTTOM
[137]   'SB_LINEUP'       W_FP THISFN VSB_LINEUP
[138]   'SB_LINEDOWN'     W_FP THISFN VSB_LINEDOWN
[139]   'SB_PAGEUP'       W_FP THISFN VSB_PAGEUP
[140]   'SB_PAGEDOWN'     W_FP THISFN VSB_PAGEDOWN
[141]   'SB_THUMBTRACK'   W_FP THISFN VSB_THUMBTRACK
[142] ''                  W_FP 'END'
[143] 'WM_HSCROLL'        W_FP 'BEGIN'
[144]   'SB_LINEUP'       W_FP THISFN HSB_LINEUP
[145]   'SB_LINEDOWN'     W_FP THISFN HSB_LINEDOWN
[146]   'SB_PAGEUP'       W_FP THISFN HSB_PAGEUP
```

```
[147]    'SB_PAGEDOWN'       W_FP THISFN HSB_PAGEDOWN
[148]    'SB_THUMBPOSITION' W_FP THISFN HSB_THUMBPOSITION
[149] '' W_FP 'END'
[150] fp←W_CreateFilter w_fp
[151] →RETURN
[152]
[153] ⍝-------------------------------------------------------------
[154] MAKE_SYSMETS_TBL:⍝ SUBROUTINE: Create System Metrics Table
[155] W_A ⊂ 'SM_CXSCREEN'            'Screen width in pixels'
[156] W_A ⊂ 'SM_CYSCREEN'            'Screen height in pixels'
[157] W_A ⊂ 'SM_CXVSCROLL'           'Vertical scroll arrow width'
[158] W_A ⊂ 'SM_CYHSCROLL'           'Horizontal scroll arrow height'
[159] W_A ⊂ 'SM_CYCAPTION'           'Caption bar height'
[160] W_A ⊂ 'SM_CXBORDER'            'Border width'
[161] W_A ⊂ 'SM_CYBORDER'            'Border height'
[162] W_A ⊂ 'SM_CXDLGFRAME'          'Dialog window frame width'
[163] W_A ⊂ 'SM_CYDLGFRAME'          'Dialog window frame height'
[164] W_A ⊂ 'SM_CYVTHUMB'            'Vertical scroll thumb width'
[165] W_A ⊂ 'SM_CXHTHUMB'            'Horzontal scroll thumb width'
[166] W_A ⊂ 'SM_CXICON'              'Icon width'
[167] W_A ⊂ 'SM_CYICON'              'Icon height'
[168] W_A ⊂ 'SM_CXCURSOR'            'Cursor width'
[169] W_A ⊂ 'SM_CYCURSOR'            'Cursor height'
[170] W_A ⊂ 'SM_CYMENU'              'Menu bar height'
[171] W_A ⊂ 'SM_CXFULLSCREEN'        'Full-screen client window width'
[172] W_A ⊂ 'SM_CYFULLSCREEN'        'Full-screen client window height'
[173] W_A ⊂ 'SM_CYKANJIWINDOW'       'Kanji window height'
[174] W_A ⊂ 'SM_MOUSEPRESENT'        'Mouse present flag'
[175] W_A ⊂ 'SM_CYVSCROLL'           'Vertical scroll arrow height'
[176] W_A ⊂ 'SM_CXHSCROLL'           'Horizontal scroll arrow width'
[177] W_A ⊂ 'SM_DEBUG'               'Debug version flag'
[178] W_A ⊂ 'SM_SWAPBUTTON'          'Mouse buttons swapped flag'
[179] W_A ⊂ 'SM_RESERVED1'           'Reserved'
[180] W_A ⊂ 'SM_RESERVED2'           'Reserved'
[181] W_A ⊂ 'SM_RESERVED3'           'Reserved'
[182] W_A ⊂ 'SM_RESERVED4'           'Reserved'
[183] W_A ⊂ 'SM_CXMIN'               'Minimum window width'
[184] W_A ⊂ 'SM_CYMIN'               'Minimum window height'
[185] W_A ⊂ 'SM_CXSIZE'              'Minimize/Maximize icon width'
[186] W_A ⊂ 'SM_CYSIZE'              'Minimize/Maximize icon height'
[187] W_A ⊂ 'SM_CXFRAME'             'Window frame width'
[188] W_A ⊂ 'SM_CYFRAME'             'Window frame height'
[189] W_A ⊂ 'SM_CXMINTRACK'          'Minimum tracking width of window'
[190] W_A ⊂ 'SM_CYMINTRACK'          'Minimum tracking height of wndow'
[191] sysmetrics←⍉ W_A ⍉
[192] →RETURN
[193]
[194] ⍝=============================================================
[195] Exit:→0
    ∇

    ∇ R←SUBR LABEL;⎕ELX;⎕IO
[1]   ⎕IO←1◊⎕ELX←'→ER' ◊ R←LABEL ◊ rtn←(⎕LC[2]+1),rtn ◊ →0
[2]   ER:rtn←,⎕LC[2]+1
    ∇
```

```
     ∇ R←RETURN;⎕ELX
[1]    ⎕ELX←'→ER' ◊ R←1↑rtn ◊ rtn←1↓rtn ◊ →0
[2]    ER:R←0 ◊ rtn←⍳0
     ∇

   ∇ R←THISFN MSG
[1]    ⍝Return string with calling function name followed by MSG
[2]    ⍝e.g.  If in MYFUN you have a line label named WM_COMMAND, and
[3]    ⍝      it is on line 123, and within MYFUN you have a
[4]    ⍝      statement like:
[5]    ⍝              X ← THISFN WM_COMMAND
[6]    ⍝      then X will contain:
[7]    ⍝              'MYFUN 123'
[8]    ⍝
[9]    ⍝ If this gets executed by the agent using the recursive style
[10]   ⍝ of writing windows programs with automatic switch statements
[11]   ⍝ you will enter the function MYFUN and branch to line 123 which
[12]   ⍝ is the line with the WM_COMMAND label.
[13]   ⍝
[14]   R←⎕SI[2;]◊R←(¯1+R⍳'[')↑R
[15]   R←R,' ',▼MSG
     ∇
```

MENTAL FLEXIBILITY TEST

Each string below contains the first letters of words that make a (sometimes!) well-known phrase. The test is to find words that make a "sensible" phrase. A "solution" to the first one is "26 Letters of the Alphabet".

A. 26 L of the A

B. 7 W of the AW

C. 1001 A N

D. 12 S of the Z

E. 54 C in a D (with J)

F. 9 P in the S S

G. 88 P K

H. 13 S on the A F

I. 32 D F at which W F

J 18 H on a G C

K. 90 D in a R A

L. 200 D for P G in M

M. 8 S on a S S

N. 3 B M (S H T R!)

O. 4 Q in a G

P. 24 H in a D

Q. 1 W on a U

R. 5 D in a Z C

S. 57 H V

T. 11 P on a F T

U. 1000 W that a P is W

V. 29 D in F in a L Y

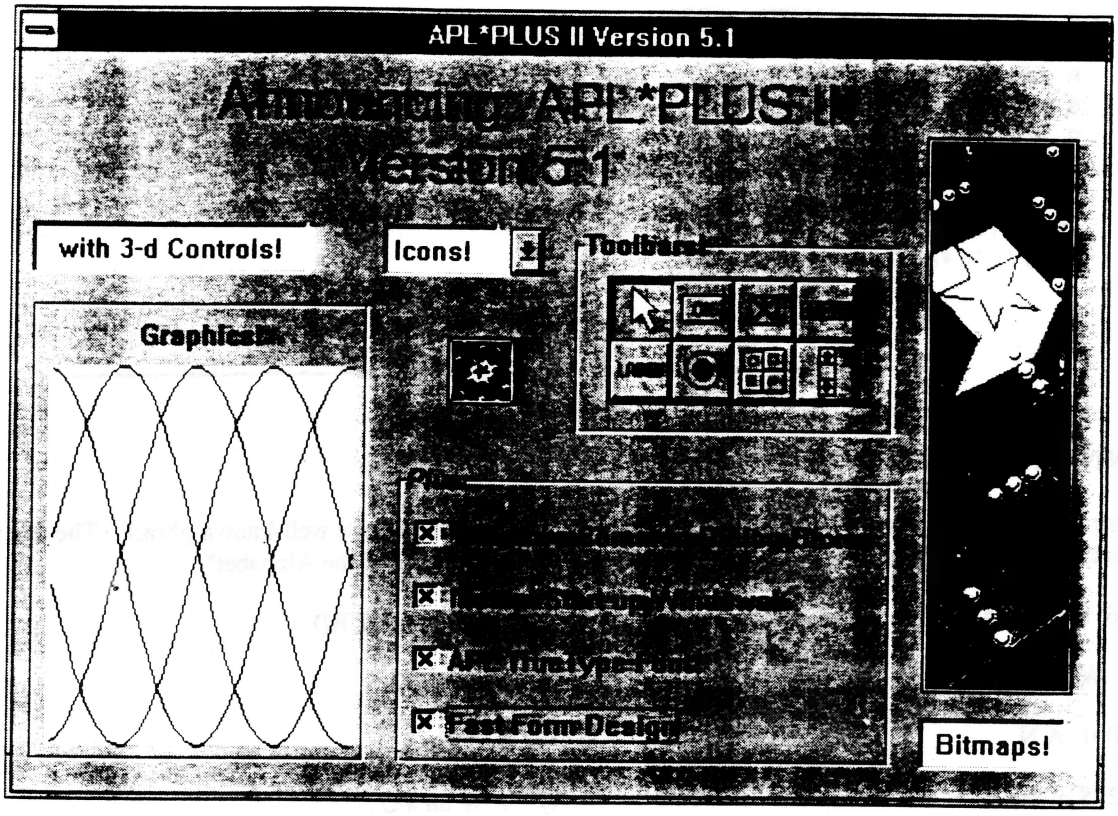
W. 64 S on a C

X. 40 D and N of the G F

Y. 4 I's in M

Z. 36 I in a Y or I in T F

We just couldn't leave well enough alone!



APL★PLUS® II for DOS and Windows™ just got better! It's now even easier (and a lot faster!) to build applications using *all* of the features of Windows 3.1. There's even a Software Developer's Kit that shows you how to *easily* add third-party custom controls!

And don't forget that **APL★PLUS II** is *the only* APL system that provides you with a fully-integrated Debugger, Paradox Interface, full DDE access, Lotus and dBase Import/Export, optimized Assembler functions, the User Command Processor, a Numeric Data Editor and the most powerful session manager available anywhere!

Get GUIng today! Call Manugistics, Inc. at **800-592-0050, ext. 625** (in Maryland, dial 301-984-5123) or Fax 301-984-5094 and get the most out of both DOS and Windows with **APL★PLUS II Version 5.1**!

APL★PLUS is a registered trademark of Manugistics, Inc., 2115 E. Jefferson Street, Rockville MD 20852.
Other trademarks are the property of their respective owners.

How to Become an APL Sysop, & Why

Dick Holt
Arlington, Virginia
e-mail: Dick.Holt@ACM.org

Advanced telecommunications are a powerful tool to bring widely scattered APL users together -- to sharpen technical skills, to get current news and market intelligence, and to accelerate productivity in APL applications.

If you or your local APL SIG would like to exploit these timely, cheap, and practical advantages, I've got an offer for you.

But first, a little background: the BBS\APL is an electronic bulletin board that I operate that provides up-to-date files, mail, and info for users of APL and J. Callers from 38 states, 6 Canadian Provinces, and 9 countries in Europe have used the BBS\APL over the past five years.

So, here's the offer: I'll provide a working copy of the BBS\APL to anyone, anywhere in the world, who agrees to run an APL BBS for at least one year. You'll get:

- All needed BBS software (Maximus, an Opus clone), FidoNet ready, and with full documentation. At first, it'll work exactly like the BBS\APL. As you learn the software, you can change it to meet your needs.
- Timesaving maintenance utilities, written in APL*PLUS (or possibly as .EXE files if you don't have APL*PLUS).
- A starter set of about 20+ megs of APL software for Manugistics, Iverson Software, IBM, Sharp, and I-APL.
- Technical assistance and support. I'll save you a lot of time getting started. You take it from there.

In 5 years of operation, I've never had a virus problem. Even so, I'll supply some parts of my anti-virus system. This BBS is immediately ready to be networked to FidoNet, and thence to the Internet, at zero or low cost.

What else is needed? Basically, you need:

- A DOS computer, dedicated full-time to the BBS (or multitasked on an existing computer). It doesn't much matter what computer you use -- an old 286 will do. A better starting point is a 386SX with an 80+ meg drive and a serial port with a 16550 UART chip. A tape backup is a big plus.
- A modem: 9600-14400 baud modems are now cheap, and modem manufacturers give good discounts to Sysops.
- A dedicated phone line. Use the cheapest service available (you won't make any outgoing calls on the BBS line).

How much time does it take? Running a BBS takes 3-6 hours/week, maybe more. You can schedule this time at your convenience. It takes time to set up a BBS the way you want it. It takes time to learn new software. I'm probably at least as busy as you are, and if I can do it, you can too -- maybe better.

What does it cost to operate a BBS? My costs -- for hardware upgrades, electricity, phone, software registration (e.g., ZMODEM), plus a lot of overhead -- add up to about $800-1200 a year. For the BBS\APL, these costs are just about covered by subscriptions, and by fees for corporate services and ads. Your costs will be different, and could easily be lower. It's important to try to make your BBS economically self sustaining.

How does this fit with the Internet? There're several factors to consider: Internet is free if your employer provides access, but perhaps as much as $10-35/month otherwise. Internet carries a zillion other useful things beyond the comp.lang.apl newsgroup. BBSs have much larger file libraries, and don't have a 7 bit problem with binary files. A BBS can echo Internet comp.lang.apl e-mail (the BBS\APL does). Commercial traffic is welcome on BBSs, but forbidden on the Internet. One size does not fit all. I use them both, each for its own unique strengths.

Why do I make this offer? First, this offer is part of the strategy that I promised in my Candidate's statement in running for the SIGAPL Board. But my motives aren't entirely altruistic. I want new sources of files and info for the BBS\APL. I need technical assistance on networking.

On a personal note, I can tell you that the BBS\APL has been extremely rewarding: I've made new friends around the country and around the world, I've strengthened my skills in several APL dialects, and (unexpectedly) improved my French by working with French APL software authors.

What do you have to do? It's easy. Just send me $US50 deposit for a BBS starter kit. I will refund this deposit, minus my postage costs to you, after you return the pile of 1.4 meg disks that I send to you. Contact Dick Holt at:

- BBS\APL: 703-528-7617, 1200-14400b, 8-N-1, 24 hours
- dick.holt@acm.org; on the Internet
- 703-528-7624 7-11pm eastern (00:00-04:00 gmt),(voice, home)
- 202-586-4449 9-5:30 eastern (14:00-22:30 gmt),(voice, work)
- 3802 N. Richmond Street, Arlington VA 22207 USA

Even an APL guru needs a vacation. Especially a working APL guru. But if you want it to be less expensive and more enjoyable, call Mike von Tiesenhausen as early in the game as possible to plan it or just to dream about it.

Sundance Travel (214) 352-4650

Personalized Service for APL gurus

TIDBITS

Manugistics' 1993 APL*PLUS Users' Conference is entitled "Solutions for the Future" and will take place September 20 and 21 in Bethesda, Maryland, USA. Registration fees are $395 ($450 after August 31, 1993) and $200 for educators and students (also $450 after August 31). To register, or for more information, contact Michele Ricard at Manugistics, Inc.; 2115 East Jefferson Street; Rockville, MD 20852-4999 USA. Telephone numbers are: 800-592-0050 (ext. 5238), 301-984-5123 in Maryland, 301-984-5412 for callers outside the USA, and 301-984-5094 (fax, attention: Michele Richard).

Manugustics also is offering APLGUI/APLDDE training September 22 - 24 in Rockville, for $950. This is a special two-part course offered in conjunction with the Users' Conference. Telephone numbers to call are the same as for the Users' Conference.

Robert Bernecky is an ACM Distinguished Lecturer. His areas of interest include language design and implementing APL systems and, currently, creating high-performance compilers for parallel systems. Bob was formerly Director of Research at I. P. Sharp Associates. He left there in 1990 to found Snake Island Research, Inc. There is a possible grand tour arrangement for Bob late this year. The objective is for Bob to speak at various APL clubs around the US, and depending on interest and costs, around the rest of the world. Eugene Mcdonnell is in charge of arranging this tour. User groups interested in hosting Bob are asked to contact Eugene at eem@pa.reuter.com to make known their interest.

USER GROUP NEWS

[This information comes from a variety of sources, mostly newsletters and attendee reports. -Ed]

Atlanta (SEAPL) Sunday, July 25, 1993, was SEAPL's Summer picnic. It was held from 2 PM to 5 PM at Stone Mountain Park. The next meeting is on September 28.

Dallas (SWAPL) The July meeting featured Todd Marshall, who presented the current version of Manugistics' new release of APL*PLUS II, version 5.1. See the review on page 13.

The August meeting will be held on Tuesday the 3rd, at 7:30 PM, at the DeVRY Institute of Technology, 4250 North Beltline Road, Irving, Texas, in Room 218. From Highway 114, go two miles south on Beltline. The building is located on the east side of the road.

The August program features Antonio Maravi, who will demonstrate his APL interpreter, which is written in C.

The September 7th program will feature APL93 attendees talking about the conference. The October 5th meeting will feature Stuart Yarus demonstrating the GUI aspects of Iverson Software's APLIWIN.

The DeVRY Institute is moving to larger quarters in November. We have been invited to move with them. Since their new facility is located near the old one, we will probably meet there, starting in November. Until then, we will continue to meet at the current DeVRY facility.

A pre-meeting dinner will be at 6 PM at the Black-Eyed Pea restaurant on Beltline Road at Northgate Road, about one mile south of DeVRY. Everyone is invited!

Northern California (APL BUG) The July 12 meeting featured David Liebtag, of IBM Corporation, talking about the beta level of APL2 for OS/2. This version is written in the C language, and will be used for all workstations. See the review beginning on page 14. Meetings are at 7 PM at the Syntex Laboratories building A2, Room 1, 3401 Hillview Avenue, Palo Alto.

There will be no August meeting.

Southern California (SOCAL-SIGAPL) The June 22 meeting featured Dr. Jim Brown of IBM Corporation. Jim talked on what IBM has been doing with APL2 for a variety of systems and platforms.

Toronto Come to APL93, August 15 - 19! The next regular meeting will be on September 27, 1993, and will include a review of APL93.

Meetings are on the fourth Monday of each month from September through May, excluding December, and are held at CIBC's 10th floor lounge in the BCE Place building, 161 Bay Street.

Washington, DC (Potomac ACM SIGAPL Chapter) On Monday, July 19, 1993, at 7:30 PM, the Potomac ACM SIGAPL, jointly with the APL Special Interest Group of the Capitol PC User Group, presented a talk on "APL Resources and the Internet".

Future Meetings: Monday, August 16, 1993 at 7:30 PM: Nancy Wheeler on APL2. Monday, September 20, 1993 at 7:30 PM: "Highlights of APL93". Monday, October 18, 1993 at 7:30 PM: Peter Lewis on AGSS.

APL SIG meetings are held at the Capitol PC User Group headquarters at 51 Monroe St., Lobby Level, Plaza East 2, Rockville, MD 20850. To find the meeting and for late information: contact PotAPL@acm.org. Late announcements and directions are also available by calling Capitol PC Users Group (CPCUG) Voice Information at (301) 762-6775.

Dallas July Meeting Notes

Margaret Brooker
Irving, Texas
email: 72124.271@CompuServe.com

Todd Marshall demonstrated the creation of a windows application using Manugistics' APL GUI for Windows. The talk was a tour through creating Windows applications culled from his own experience. He showed us two applications. One was a mapping program that drew a map of Texas and displayed various towns in the state. He showed us how Windows uses object names created using the "Hungarian notation" and compound naming. Todd is a personal proponent of the "Danish-Hungarian" notation.

He started by showing us a Windows application he has built using the CIA World Database which is about 2.5 Megabytes of raw data. He has created a Windows screen as an interface to the data which lets you select and display the information based on any of several categories. For each application he showed us how he built the forms, the tools that he used, and the properties of the objects. It is possible to generate a form visually using the supplied tools and then go into APL and fine tune the form by changing the APL code that has been generated. This allows a certain amount of spontaneity in creating the application. Todd finds that it is easy to develop the application by trial and error rather than starting from a set design. After all, APL is the ideal prototyping language! He showed us the complete code for each application, which was impressive by its consciseness.

APL User Groups (United States)

The following is a listing of all known cities in the United States with active APL user groups, along with the name of a person to contact and at least one way to contact him/her. Not all user groups have newsletters, but all do meet periodically. If you know of any additional groups (that would like to be listed), please let the editor know at his masthead address. Updates to contact information are most welcome.

Atlanta, Georgia
Doc Manges
991 Creekdale Drive
Clarkston, GA 30021
(404) 292-4968

Chicago, Illinois
Larry Mysz
836 Highland Drive
Chicago Heights, IL 60411

Dallas, Texas
Margaret Brooker
P. O. Box 210367
Bedford, Texas 76095
(817) 577-0165
CompuServe: 72124,271
Internet: 72124.271@compuserve.com

Hartford, Connecticut
Bob Pomeroy
Mass Mutual Life
1295 State Street
Mail Drop F465
Springfield, MA 01111
(413) 788-8411, ext. 2838

New York, New York
Jacob Brickman
P. O. Box 138
New York, NY 10185-0002
(718) 773-4093

Northern California
Curtis A. Jones
228 South 15th Street
San Jose, CA 95112-2150
Internet: jonesca@vnet.ibm.com
Prodigy: CJGH56A
FidoNet: 1:143/11

Southern California
Roy A. Sykes, Jr.
Sykes Systems, Inc.
4649 Willens Avenue
Woodland Hills, CA 91364-3812
(818) 222-2759

Washington, District of Columbia
John A. Martin
(301) 497-2698
fax: (301) 498-8260
Internet: jmartin@milo.starlab.csc.com

APL2 FOR OS/2

Rick Butterworth
Monterey, California
e-mail: 71420.742@CompuServe.com

If you are an OS/2® user, this should interest you; if you are not an OS/2 user, this might interest you in becoming one; if you are an APL2 user, this will definitely interest you. The APL Bay Area Users' Group, APL BUG for short, hosted David Liebtag for their July meeting. David is in the APL2 development group at IBM's Santa Teresa Laboratory. He presented the beta version of APL2 for OS/2, Version 2.x, in a wide-ranging, 2 hour discussion and demonstration. IBM is meeting its commitment to implement a common APL2 language on all supported platforms, with compatible auxiliary processors (APs). The following sneak preview might be of interest to all APLers.

The Basics

The APL2 source is now in 'C', and is common to the RISC System/6000 and an upcoming SUN implementation, thus easing maintainability and compatibility. The session manager and other operating system specific support are written specifically for OS/2, Version 2.x; this APL2 system is a native OS/2 application.

The APL development group has integrated APL2 into OS/2. Multi-tasking is an OS/2 strength, and APL2 builds on this. For example, the session manager is a separate task from the interpreter itself. The editor is yet another task, and when application code fires up, that's yet another task. Auxiliary Processors also run as separate tasks. These tasks occur asynchronously, so control returns to the user as soon as possible. It all plays together quite well, a credit to OS/2 and APL2's integration.

Printing is an example of OS/2 integration. We saw a PM (Presentation Manager) window holding print jobs spawned by the APL session. From that window, the user can preview, route, hold, or discard printer output. Any OS/2 supported printer is an APL printer, no problem. OS/2 also provides font support. We watched while David reset the session font from an OS/2 window. APL2 uses any OS/2 font and IBM will deliver an APL Adobe PostScript Type 1 font for the initial release.

Keyboard support for APL2 is specific to APL2, and uses OS/2 dialogs to select or re-map the keyboard. David brought up a PM window illustrating the current character to keyboard assignments. There were several keyboards, accessed by clicking on an icon, and David created another keyboard mapping on the fly.

Cooperative Processing

IBM has held out *cooperative processing* as an intelligent and powerful alternative to the popular notion of down-sizing, among other things. The idea is to design an application with clean interfaces between the

major areas of common activity, specifically with shared variables. Networking between workstations and mainframes, the IBM user distributes each portion of the application to the appropriate platform. Once a shared variable is initialized, the target platform or process is transparent to the application.

For example, a database application might partition itself into a database engine, computation, and user interface. When all functions are mainframe resident, the user interface options are relatively unattractive by today's standards. A workstation however is not a smart place to put data to which many users need access to a common copy. Through cooperative processing, the user interface can exploit the workstation graphical user interface and windowing capability. The computation can reside on the platform with suitable power and availability. The database can remain on a centrally located mainframe where a single copy is subject to quality control. Each application module runs where it makes the most sense. The cooperative processing idea is much easier to describe than carry out.

IBM has been building the cooperative processing infrastructure, relying on TCP/IP as the prerequisite network protocol. Some 15 month's ago, APL2 for VM and TSO implemented cooperative processing using shared variables and auxiliary processors. With APL2 for OS/2, the PC workstation falls into place. APL2 for the RISC System/6000 is due for the update soon, as will a new APL2 product for the SUN workstation. All APL2 platforms, except the PC under DOS, will soon have virtually identical facilities for cooperative processing. (The APL2 for PC product is not likely to, since that would entail running TCP/IP under DOS.)

David demonstrated the concept by starting two APL2 sessions under OS/2 on the same machine and sharing a variable between them. With no apparent difficulty, data was moving between the two sessions. The windows were side by side on the same machine, but they might have been on two machines, one a remote mainframe, with no syntactic difference in the interface mechanism. Cooperative processing is a powerful concept and APL2 for OS/2 completes another link.

Performance

There were many questions about the performance of APL2 under OS/2. David confessed that it was not yet up to his own expectations, but speculated that tuning efforts would improve the situation prior to release. Overall, he expected performance to be on a par with the DOS APL2 product, which is really saying something, considering the greater complexity of OS/2 and how much more is going on. His demonstration machine was a 486/50 with 20 megabytes of RAM. Things looked a little sluggish for that machine, but there was no hard data on this point.

We asked David how much memory a machine needed to run APL2 under OS/2. He recommended 16 megabytes, but Jim Brown (IBM's APL Chief Architect) told me later that he routinely uses an eight megabyte machine with no problem.

Editor and Session Manager

The session editor is a classic ring editor using OS/2 cut, copy, and paste functions. The session manager starts an edit session when a name in the session log has been double clicked, and the editor can be called under program control. There were no line numbers on functions in the editor, touted as a feature. It was remarked that this should reduce the occurrence of branches to absolute line numbers. I actually found function displays somewhat stark without line numbers and would have preferred optional line numbers.

The session manager maintains the user log as a perpetual file to which you are forever appending, like the other APL2 implementations. The session manager acts somewhat like a log editor, with cut and paste, search and replace. The system classifies log text into 16 distinct categories, for example quad input, quote-quad input, system error output, etc. The user may assign each one its own color, leading to a real rainbow of possibilities.

The session manager is smart about managing the display. David executed `⍳10000`, and we just saw the last screen. The entire display was in the log, but we were spared the inconvenience of watching it scroll by. This behavior is very handy, and the user may turn it on or off.

There is context sensitive on-line help for the session manager. David assured us that it was quite comprehensive. Function key F1 offers universal access to the help facility, implemented using the OS/2 help mechanism. They are working on the language reference, but that will not be on-line in the first release.

Auxiliary Processors, Shared Variables, and *⎕NA*

Someone asked David to explain the difference between *auxiliary processors* (AP's) and *⎕NA*. It was a good question. Shared variables access AP's which are intrinsically asynchronous. (AP's may choose to behave synchronously.) This means that the shared variable mechanism returns control after initiation of an AP

request. If you check for completion however, and the AP has not completed its task, the shared variable mechanism holds until a return code is available. By contrast, the □*NA* facility loads code from outside APL and executes it. In this sense, David explained that it was more like a subroutine call. APL waits for the call to return before releasing control, hence it's synchronous.

APL2 comes with ten AP's and a 'C' source template for creating your own. They cover the common core of APL2 AP's, and some specifically for the OS/2 environment. Here is the list:

AP 100 OS/2 Command Processor
AP 101 Stack Processor
AP 124 Text based full screen forms
AP 119 Low Level TCP/IP ("C Socket Level')
AP 120 Session Manager
AP 127 DB2/SQL Processor
AP 145 Presentation Manager API
AP 207 Draw, Fill, Write Level Graphics
AP 210 Record based File I/O
AP 211 Object Library file system

Dave demonstrated most of these AP's, however AP 145 drew the most interest; this is the mechanism for building PM applications. Dave showed us a handful of supplied functions providing common dialogs and demonstrating use of AP 145. These examples included selecting items from a list, posting a screen message, and requesting text. In his own words, David said "This (AP 145) is a low level interface. Any OS/2 API call you can make from 'C' is callable from APL using AP 145." David stressed that, thanks to AP145, the APL2 programmer can create multi-threaded processes. The implication for APL application builders however remained just a bit obscure.

This is of course both the good news and the bad news. Dave showed us the classic calculator, an APL2 application, accepting complex numbers no less, running as a PM window. It was a 100 line cryptic function relying on a DLL that David built separately using the OS/2 developers tool kit. Creating dialogs in APL2 from scratch are practical, but according to David, are not as easy as he would like. The preferred method is using the OS/2 developers tool kit, especially the resource compiler for building DLL's, and then using the DLL from AP145. I would assume the OS/2 developers tool kit, an extra cost item, is necessary for its documentation at the very least to make good use of AP 145. Assuming the OS/2 API is at least as complex as the Windows API, it's a safe bet that few APL2 users will conquer this interface anytime soon.

David explained that they were looking at ways to add a high-level interface to the OS/2 API. They recognized the competition had high level access to the Windows API, and felt APL2 should provide a comparable capability. It doesn't sound like it will happen with the first release however. I asked if there were plans for bringing up APL2 under Windows, and the answer was yes, they would like to, after meeting the goals for OS/2.

Bits and Pieces

IBM has not set the price for APL2 for OS/2 and apparently there is intense discussion within IBM on this topic. A formal product announcement is not planned for APL93; however, IBM will demonstrate the beta version of APL2 for OS/2 and other platforms at that time. The final product is not yet in concrete and we could see that they were still making tradeoffs in its final configuration. Also in the works are updates to APL2 for VM and TSO, APL2 for the RISC System 6000, and the initiation of APL2 for SUN workstations.

David boasted that APL is enjoying an unprecedented level of acceptance in the IBM front office as the model product line. Besides said David, IBM could never abandon APL; the single biggest corporate user is IBM.

APL*PLUS II GUI Reference Card

Todd B. Marshall
Conroe, Texas
e-mail: 70544.715@CompuServe.com

Anyone who has worked with Windows and simplifications thereof, like Microsoft's Visual Basic and Manugistics' APLGUI, knows they are mechanisms of many knobs. The knobs have many contexts and many settings.

Having made three or four iterations through the volumes of information, you are probably ready for summarization, encapsulation, and organization. Working with Windows and Manugistics' APLGUI, I discovered that a concise, ready reference was needed and unavailable.

The following is just such a reference which I have compiled for the APLGUI product. I have found it to be a ready, convenient, and timesaving resource for working with the product.

APLGUI Reference Card:

PROPERTIES:

border:
0-None
1-Thin
2-Thick (sizable)
3-Modal
4-System Modal
16-System Menu
32-Minimize Box
64-Maximize Box

color:
positive: Red Green Blue (0-255)
-1 Object:Color of Form
-2 Windows for this class
-100 Scroll Bar
-101 Background
-102 Active Caption
-103 Inactive Caption
-104 Menu
-105 Window
-106 Window Frame
-107 Menu Text
-108 Window Text
-109 Caption Text
-110 Active Border
-111 Inactive Border
-112 Application Workspace
-113 Highlight Background
-114 Highlight Text
-115 Button Face
-116 Button Shadow
-117 Gray Text
-118 Button Text
-119 Caption Text
-120 Button Highlight

font
[1] FontName
WAPLFONT for APL w/oem
Select Edit style 2048
[2] FontSize
[3] Fontstyle:
1-Bold
2-Italic
4-Underline
8-Strikeout
16-Opaque
pitch:
0-Default
32-Fixed pitch
64-variable pitch
128*family
family:
0-Don't care
128-Roman
256-Swiss
384-Modern
512-Script
640-Decorative
quality:
0-Default
1024-Draft
2048-Proof
[4] FontCharset
ansi (default)
oem (WAPLFONT)
symbol

limit
Edit/Combo
0-no limit
n-number of chars
List
Empty-No Horz Scroll
[1] Opt: width of list/cols
[2] Opt: number of cols

list
List , Combo, or Toolbox objects

mode
[1] Transparency Mode
0-Transparent
1-Opaque
[2] RasterOp Mode
1-Black
2-Not Merge Pen
3-Mask Not Pen
4-Not Copy Pen
5-Mask Pen Not
6-Not
7-XOR Pen
9-Not Mask Pen
10-Not XOR Pen
11-Nop
12-Merge Not Pen
13-Copy Pen (default)
14-Merge Pen Not
15-Merge Pen
16-White
[3] Poly Fill Mode
0-Alternate (default)
1-Winding

pointer
0 determined by control or windows (default)
1-Arrow
2-Cross-hair
3-I-Beam
4-Icon
5-Size(4 points)
6-Size(NE SW)
7-Size(NS)
8-Size(NW SE)
9-Size(WE)
10-Up Arrow

scale
Element[1]
0 User defined
1 Characters
2 Dialog units
3-Points
4-Twips (1440/inch)
5-Pixels
Elements[2-5] row, column, height, width

style
Button
1-default button
2-cancel button
CheckBox
style
0-Box left of text
1-Box right of text
ComboBox
0 DropDown
1 Simple (always down)
2 DropDownList (no edit)
8 Inhibit auto horz scroll
16 Vert scroll bar
128 Hide unnecessary scroll
Edit
4-Multi-line
8-Inhibit auto vert scroll
16-Vert scroll bar
32-Inhibit auto horz scroll
64-Horz scroll bar
128-Password
256-Lowercase
512-Uppercase
1024-Don't hide when tabbing out
2048-APL entry
4096-Read only
8192-Multiline incorporate Enter
Frame
0-Standard
1-Inset
2-Level
3-Outset
Label
0-Left Justify
1-Centered
2-Right Justify
4-Inhibit & Prefix
8-Inhibit word wrap
ListBox
2-Multiple selections
2-Display vert fractional line
16-Vertical scroll bar
32-Multiple cols (limit)
64-Horz scroll bar (limit)
128-Hide unnecessary scroll bar

Menu
0-Not toggle item
1-Toggle item
OptionButton
0-Radio button left
1-Radio button right
Scroll
0-Vertical
1-Horizontal
2-Size box
0-No Alignment
4-Bottom/right
8-Top/left
16-Track thumb dragging

tab
Not form or Menu
Tab path thru form
Neg to group options

text
Edit/Combo

value
Button: 1-Down, 0-Up, -1 flash
Check: 1-Checked, 0-Not checked
Combo: 0-no selection, n-selection no.
List: vector of current choices
Menu: 1-checked,0-unchecked
Option: 1-checked,0-unchecked
Scroll:Current, Max, PgSize, Min, Tracking

METHODS:

Draw
Drawing Renderings
Bitmap
[1] bitmap
[2,3,4,5] r,c,h w
Circle
[1-3] r,c,radius
[4-7] optional: aspect,start end,border
border values:
0-No border
1-Circle border (default)
2-Start Angle border
4-End angle border
Clear
Icon
[1,2,3]icon,r,c
Line
[1...n] points
Point
[1,2] r,c
[3] Opt: color (r,g,b)
Poly
[1...n] points
Rect
[1-4] r,c,h,w
[5,6] Opt: eliptical h,w corners
Text
[1-3] text,r,c
[4,5] Opt: clip area: h,w

Draw
Drawing States:
Brush
[1] style
[2] Opt: Color (r,g,b)
[1,2] -1 hbrush
[1,2] -2 bitmap
Brush style
1-Solid (default)
2-Horz hatch
3-Vert hatch
4-Up diagonal
5-Down diagonal
6-Cross hatch
7-Diag. Cross Hatch
Color
[1] Color Back (r,g,b)
[2] Opt: Color Text (r,g,b)
Font
[1] name
[2-6] Opt: size,style,charset,angle
(see font property)
Mode
[1] Transparency
0-Transparent, 1-Opaque
[2] Raster Op
1-Black
2-Not Merge Pen
3-Mask Not Pen
4-Not Copy Pen
5-Mask Pen Not
6-Not
7-XOR Pen
9-Not Mask Pen
10-Not XOR Pen
11-Nop
12-Merge Not Pen
13-Copy Pen (default)
14-Merge Pen Not
15-Merge Pen
16-White
[3] PollyFillMode
0-Alternate (default)
1-Winding
Pen
[1] style
-1 handle
0 Transparent
1 Solid (default)
2 Solid inside
3 Dash
4 Dot
5 Dash-dot
6 Dash-dot-dot
[2] Opt: color (r,g,b)
[3] Opt: size width in pixels

Hide
visible
Before,After Hide
3,-3 Maximized,Maximized hidden
2,-2 Minimized,Minimized hidden
1,-1 Normal size,Normal size hidden
0,0 Just created,not yet visible
-1,-1 Normal size, Normal size hidden
-2,-2 Minimized, Minimized hidden
-3,-3 Maximized, Maximized hidden

Show
visible
Before,After Show
3,3 Maximized,Maximized
2,2 Minimized,Minimized
1,1 Normal
0,1 Just created not visible,Normal
-1,1 Normal hidden,normal
-2,2 Minimized hidden, Minimized
-3,3 Maximized hidden, Maximized

CLASSES of OBJECTS:

Button (bn)
style:
1-Default
2-Cancel

Check (ck)
style:
0-Box on left
1-Box on right
value:
1-Checked
0-Unchecked

Combo (cb)
style:
0- Drop Down
1- Simple (always down)
2- DropDown List (no edit)
8- Inhibit auto horz scroll
16-Vert scroll bar in list
128-Hide scroll bar not necessary

Edit (ed)
style:
4-Multi-line
8-Inhibit auto vertical scrolling (4)
16-Vert Scroll bar (4)
32-Inhibit auto horz scroll
64-Horz scroll bar
128-Password
256-Lower case (not 512)

512-Uppercase (not 256)
1024-Don't hide selection when tabbing out
2048-APL entry field
4096-Read only
8192-Enter chars incorporated (4)

Frame (f)
style:
0-Standard
1-Inset
2-Level
3-Outset
mode:
[1] Transparency
[2] Draw
[3] Fill

Label (l)
style:
0-Left Justified
1-Centered
2-Right Justified
4-Inhibit & Prefix
8-Inhibit word wrap

List (ls)
style:
1-Multiple Selections
2-Display vert fractional line
16-Verticle scroll bar
32-Multiple columns
64-Horizontal scroll bar
128-Hide scroll (short list)

Option (op)
style:
0-button left of text
1-button right of text

Printer ()

Scroll (sb)
style
0 Vertical
1 Horizontal
2 Sizebox
0 No alignment
4 Bottom/right
8 Top/left
16 Track thumb dragging
value
[1] current pos
[2] max pos (def 100)
[3] page size increment (def 1)
[4] min pos (def 0)
[5] enable tracking thumb

ToolBox (tb)
list:
[1] Button Up Icon/BM
[2] Button Down Icon/BM
[3] Button Sel Icon/BM
mode:
0-Disabled
1-Pushbutton
2-Toggle
3-Option
4_Option Mode 2/ Primary Cell
style:
0-Icon
1-Bitmap
0-No overlap or spacing
16-One Pixel spacing
32-One Pixel overlap
value:
Icon/BM presented (0,1,or 2)

EVENTS:

KeyDown, KeyPress, KeyUp
wArg[1] key scan code
wArg[2] repeat count
wArg[3]
1-Shift Key Down
2-Ctrl Key Down
4-Alt Key Down
wArg[4]
1-Shift key toggled
2-Ctrl key toggled
4-Alt key toggled
wArg[5]=1 extend key such as arrow
wArg[6]=1 key is repeated
KeyPress return
wZ =0 Pass on, -1 Ignore, -2 reject with beep. Translate by setting new value

Mouse (ToolBox only)
1-Press
2-Drag off
3-Drag on
4-Release
5-Click
6-Double Click

MouseDown
wArg[1 2] row,column coordinates
wArg[3] Pressed 1-Left;2-Right;4-Middle
wArg[4] Down 1-Left;2-Right;4-Middle
wArg[5] Shift Down 1-Shift;2-Ctrl;4-Alt
wZ 0-system passes control (default)
wZ -1 system doesnt pass control
wZ -2 -1 with beep

MouseDrag
wArg[1 2] row,column coordinates
wArg[3] 0
wArg[4] Down 1-Left;2-Right;4-Middle
wArg[5] Shift 1-Shift;2-Ctrl;4-Alt
wZ
-1-system passes event,following not sent
0-system passes event
-1 system doesn't pass event
-2 -1 with beep

MouseMove
wArg[1 2] row,column coordinates
wArg[3] 0
wArg[4] State 1-Left,2-Right,4-Middle
wArg[5] Shift 1-Down,2-Ctrl,4-Middle
wZ
0-system passes event
1-system doesn't pass event
2-1 with beep

Resize
0-Resized or restored or being created
1-Minimized
2-Maximized

USER FUNCTIONS:

result <-- *[options]* WChooseColor *initc [col1 col2 ... col16]:*

initc: RGB color initially selected (0=black)
col1...col16: RGB values of up to 16 colors
options: 2-Display custom colors; 4-Disallow custom colors
result: 0 if exit without selection; else RGB color selected followed by custom colors

result <--*[options[device]]* WChooseFont *logfont color:*

logfont: 14 element nested array defining font
[1] height
[2] width
[3] escapement
[4] orientation
[5] weight
[6] italic
[7] underline
[8] strikeout
[9] char set
[10] out precision
[11] clip precision
[12] quality
[13] pitch and family
[14] face name

color: RGB color displayed in color box when form presented (0 for black)
options:
1-screen fonts only
2-fonts for *device*
256-enable strikeout, underline, and color
1024-Windows sets only
2048-No vector fonts
4096-No GDI font simulations
16384-Only monospace fonts
32768-Only printer and screen fonts
65536-Error on unknow select
131072-Only Scalable fonts
252144-Only TrueType fonts
device: 0 for display; else WPrintDlg result or other device context
result: 2 item array. [1]14 element logfont array; [2] RGB font color

result<-- ***[options]*** **WGetOpenFileName** ***dir file ext title filters fi ro:***

result<-- ***[options]*** **WGetSaveFileName** ***dir file ext title filters fi ro:***

dir,file,ext: directory, file and extension initially displayed
title: Caption for dialog box
filters: ***(descn\|specn) pairs:*** *descn:* description of filter (e.g. Docs *.txt;*.doc)
specn: Specification (e.g *.txt; *.doc
Default is All Files '(*.*)|*.*'
fi: file filter user initially selects (default is 1)
ro: read only check box
options:
4-Hide read-only check box
256-Allow invalid characters
512-Allow multiple selection
2048-Allow only valid paths
4096-Allow only existing files
8192-Notify if non-existant
32768-Allows not read-only
result:
[1] directory selected
[2] file(s) selected (blank delimited)
[3] same as input extension
[4] same as input title
[5] same as input filters
[6] index of filter selected
[7] setting of read-only check box

result <-- ***[options[wHwnd]]*** **Whelp** ***file topic (or file context)***

file: Help file to display
topic: Topic to display
context: Context to display
options: behavior
1-help for specified context
2-shuts down help system
3-displays index
4-displays help for help
5-sets current index to context
257-displays help for topic
261-displays list of topics
wHwnd: window handle requesting help (default 0)
result: 1 if successful; else 0

handle <-- Whwnd:

Returns window handle

filename **WimportForm** ***new-formname:***

Import dialog box and control objects from resource file and define APLGUI form in workspace

z <-- ***[self]*** **Win** ***winarg [arguments]:***

self: object handle overriding default
winarg:
.feature
:feature
control.feature
form:feature
form:control.feature
form[i]:control[j].feature
form[i]control[].feature
form[i]:control.feature
form:control[j].feature
form:control[].feature
control[j].feature
control[].feature
form[i]:feature
form[]:feature
feature: name of method or property
control: name of control on current form
form: name of form
i: instance of named form
j: instance of named object
arguments: If feature is property, arguments contain new property value. If feature is method, arguments are stored in *wArg* global.

rc <--*[sep]* **Wmsg** ***msg, caption, ctrl:***

ctrl:
0-Ok
1-Ok Cancel
2-Abort Retry Cancel
3-Yes No Cancel
4-Yes No
5-Retry Cancel
0-No icon
16-Stop Sign
32-Question mark
48-Exclamation mark
64-Asterisk
0-First push button
256-Second push button
512-Third push button
rc:
1-Ok
2-Cancel
3-Abort
4-Retry
5-Ignore
6-Yes
7-No

result<-- ***[options]*** **WPrintDlg** ***[hdc range from to min max copies tofile colate] or ['']:***

hdc printer device context or 0*:*
range: 0-All, 1-Selection, 2-Pages
from: initial from page
to: initial to page
min: lowest page number
max: highest page number
copies: number of copies
tofile: 0 Unchecked; 1-Checked
collate: 0-Unchecked; 1-Checked
options:
4 Disable Selection
8 Disable Pages
64 Display Print Setup dialog box
256 Return device context handle
524288 Disable print to file check
1048576 Hide print to file check
result:
[1] 0 or device context (opt 256)
[2] range (0-All,1-Sel,2-Pages)
[3] from value
[4] to value
[5] Min
[6] Max
[7] Copies
[8] Print to file check
[9] Collate copies check

index <-- **Wtoolbox** ***row, column, event:***

row,column: row and column of tool box to select
event: -1 to unselect cell

-- NOTICE --

(for those who missed this ad last month)

Kathy Parker (formerly with Decision Consultants, Inc.),

announces the opening of new offices:

Mail: Kathy Parker & Associates
14232 Marsh Lane, Suite # 343
Dallas, Texas 75234

Office: 214-386-0293 (voice)
214-386-0292 (fax)

(Voice mail messages and faxes are also retrieved after regular business hours and on weekends.)

Specializing in contract and permanent positions in:

APL/APL2	SAP
C, with DOS, OS/2, or Unix	Unisys 1100/2200
Oracle	IEF
C++	AS/400
PowerBuilder	MS and X Windows
Artificial Intelligence	Open Systems
Sybase	Informix
LAN/WAN/MAN	Banking/Financial
Documentation	Information Development
MSA / D & B	Hogan
Project Management	Foundation CASE
Cobol, CICS	OSF/DCE & DME
Client/Server	SNA
CICS Systems	LU 6.2

Positions open in many major cities for candidates with a minimum of one year's industry experience.

Call for current and future openings; mail or fax resume in strictest confidence.

GUIDELINES FOR DESIGNING AND PROGRAMMING GRAPHICAL USER INTERFACES

Marc Griffiths
Toronto, Canada
e-mail: marcg@utcc.utoronto.ca
e-mail: 76260.3314@CompuServe.com

Graphical user interface (GUI) software for personal computers provides a multitude of features which present opportunities and challenges for designing user interfaces as well as entire application systems. In this article, I intend to limit my remarks to the former, but I encourage others to share ideas on how to structure GUI applications and how to avoid undesirable practices. I hope to provide a brief guide to some sources of useful information which will help to minimize both the learning curve and the number of serious interface design flaws as APL applications are written for GUI environments. In future articles, I shall look at specific aspects of interface design in detail.

This article stems from several observations. Windowing technology is far from new: the Apple Macintosh and the first version of Microsoft Windows were released in the mid-1980s, and were themselves descended from earlier work on the Xerox Star and Apple Lisa. The speed and directions of research into, and development of, user interfaces continues to accelerate and broaden. One need only look at the January, April and July issues of the Communications of the ACM this year for collections of articles on the themes of multimedia, the next generation of graphical user interfaces, and computer augmented environments. If we -- the APL community -- don't move quickly, we'll be left in the dust, scrambling for a decreasing number of isolated pockets of APL activity. The real tragedy, in my view, is the prospect of the world at large perceiving APL as obsolete because it is incapable of taking advantage of new operating systems and interfaces at a time when many other features of APL system development -- prototyping, evolutionary system design, and participatory design -- are gaining in legitimacy and importance. (The theme of the June 1993 issue of the Communications of the ACM was participatory design.)

Within this broader context, we need only look at recent issues of, for example, *Vector* and the *SWAPL Newsletter*, to see the extent of the APL community's pre-occupation with the APL interpreters from Dyadic, Iverson Software, and Manugistics which run under Microsoft Windows. (These come in addition to the long-standing MicroAPL offerings for the Macintosh, the Commodore Amiga, and the Atari ST.) I share the authors' enthusiasm for the products, as well as their optimism for APL's prospects as a medium for delivering fully functioning Windows applications. But before we rush headlong into the fray, now is an opportune moment to take a step back and look a bit more closely at just what the APL community is about to become a part of, and what some of the implications are in terms of designing systems. As latecomers to this arena, we need to take stock of the precedents which have been set and where things stand now before trying to make our mark on it.

In very simple terms, a graphical user interface is one in which several different application programs appear to be available simultaneously. All applications exhibit common and predictable visual and behavioural characteristics, and each allows the user to interact with it by means of different devices (keyboard, pointing devices, etc.) according to certain established conventions. This is not to say that such environments prohibit the design and implementation of original, creative application programs. If anything they offer far too many opportunities to do so, and to go badly wrong in the process. The important point to bear in mind is that several million users have had their expectations moulded, and their actions conditioned, by application software which they are running. If a new application is to fit seamlessly into one of these environments, it must conform. When it deviates from the standard, it should do so for a very good reason and in an acceptable way.

The five books listed at the conclusion of this article fall into two categories. The vendor-supplied texts from Apple, IBM and Microsoft provide guidelines for designing interfaces. Their treatment of the user interface is somewhat general in trying to accommodate all types of applications while, at the same time, being specific to the graphical user interface software supplied by each company (Macintosh and Apple II operating systems, OS/2, and Windows respectively). The introductory portion of each text is given over to the philosophy, principles, concepts, and methods underlying interface design. The extent and quality of these, and of the associated bibliographies, is highly variable: Microsoft is miserly, Apple adequate, and IBM by far the most thoroughly explained and illustrated. The greater portion of each volume is devoted to enumerating and explaining the conventions which have been adopted with respect to the appearance and behaviour of standard windows, dialog boxes, menus and controls, as well as listings of function keys, keyboard equivalents to menu items and so on. IBM's Common User Access (CUA) Guidelines are the most thorough and clearly expressed. Neither Macintosh nor Windows applications necessarily

conform to the CUA recommendations, so that being aware of one set of rules does not necessarily help with situations involving a different platform. However, the three volumes allow the reader to assess the possible effects of design decisions in the user interface of an application which will need to run in more than one of these environments.

All three of these books acknowledge that user interface design is only one area within the broader realm of Human-Computer Interaction (HCI or CHI). The two books by Mayhew and Schneiderman address the broader issues of HCI in terms of current research and development, as well as providing recommendations for user interface designers. The Mayhew volume is likely to be of more immediate use to practitioners. Schneiderman provides more historical background and casts his net wider to include new hardware devices, special considerations for users with disabilities, etc. In both cases, the discussions are not limited to graphical user interfaces, but extend from command-line interface design at one extreme, to highly sophisticated input-output devices which utilize gesture and speech at the other. Both of these texts emphasize the importance of the user and the need for numerous iterations when designing the user interface, treating these issues in the context of short- and long-term memory, cognitive styles, task analysis, and the conflicting demands of ease-of-use versus ease-of-learning. More specific discussions focus on the use of colour, the design and use of icons, screen layouts, error handling, on-line help, and user manuals.

Each vendor's guidelines describes the "what" and, to a lesser extent, the "how" of interface design for its own graphical user interface software and is, therefore, a prerequisite for anyone planning to write applications for any of these environments. Of these, the Microsoft volume is the least satisfactory and provides too little support on its own for effective design. The Mayhew and Schneiderman books consider guidelines in terms of a variety of software environments and do more to address the "why" or, minimally, the "why not" behind the recommended designs and practices. Each of these is a highly desirable supplement to any of the vendors' guidelines. No one of these purports to be a graphical user interface cookbook to be followed slavishly. As most APL applications programmers know, users and user environments differ widely, and adapting recommended practices to specific circumstances is the key to designing successful systems.

Vendor Guidelines:

Apple Computer Incorporated (1987) **Human Interface Guidelines: The Apple Desktop Interface**. Reading, MA: Addison-Wesley. 144 pages.

International Business Machines Corporation (1992) **Object-Oriented Interface Design: Common User Access Guidelines**. Carmel, IN: Que Corp. 708 pages.

Microsoft Corporation (1992) **The Windows Interface: An Application Design Guide**. Redmond, WA: Microsoft Press. 228 pages.

HCI References:

Mayhew, Deborah J. (1992) **Principles and Guidelines in Software User Interface Design**. Englewood Cliffs, NJ: Prentice-Hall. 619 pages.

Schneiderman, Ben. (1992) **Designing the User Interface: Strategies for Effective Human-Computer Interaction** (2d ed.). Reading, MA: Addison-Wesley. 573 pages.

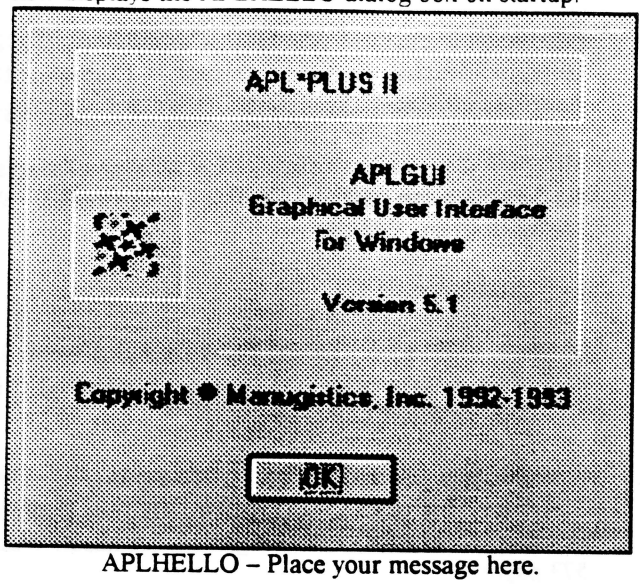
APPLICATION NOTES

Rick Butterworth
Monterey, California
e-mail: 71420,742@CompuServe.com

By now, isn't everybody up on Manugistics® Version 5.1 of APL*PLUS® II? Well, when you see how easy it is to get professional results with Version 5.1, you will be soon. This is the fifth in a series on APLGUI, the APL*PLUS II high-level interface to Windows™ 3.1. Column 0 (that `⎕IO` thing again) covered APLGUI installation and several demo forms. Column 1 discussed `Wed`, the "point and click" forms editor for building APLGUI forms. Column 2 examined the support function `WGetOpenFileName` and its twin sister `WGetSaveFileName`. These support functions tap existing Windows dialogs for enhanced productivity and reliability, and immediate acceptance as a "Windows" application. Finally, column 3 discussed control object instances, a powerful concept in form building. This month's column will cover several more support functions in this category. This column will concentrate on the Version 5.1 upgrade released in May. The basics are all in place. There is no longer any reason to hesitate.

APLHELLO

We have all seen the opening dialog box in Windows applications displaying the application's title, the version number, and the obligatory copyright notice. APLGUI now displays the APLHELLO dialog box on startup.



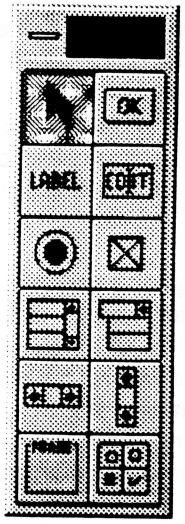
APLHELLO – Place your message here.

This is APLHELLO, a dialog box compiled into APLGUI.EXE. APLGUI.EXE displays the dialog before APL.EXE is launched, so it appears quickly while APL is getting organized. Using a resource editor such as Borland's Resource Workshop or Microsoft's Visual C++ Application (APP) Studio, you can edit this resource to suit your application and taste. I haven't tried this, but it should be straightforward. To close the dialog under program control, execute `CloseHello`, found in the WINDOWS workspace. To suppress the display, set HELLO=0 in APLGUI.INI with any text editor. On startup the dialog just doesn't appear. An alternative procedure involves Microsoft's Software Developer's Kit (SDK) and the file APLGUI.RC. In any event, it lends a touch of class to your APLGUI application.

`Wed` Editor

This "point and click" editor has gotten better yet with version 5.1. For me, the best new `Wed` feature is the alignment grid on the form's client area. The grid is calibrated in the form's scaling units, and is adjustable in both horizontal and vertical directions, a nice touch. You can cause controls to "snap to" the grid, or just use it for visual alignment. It is much easier to create perfectly aligned controls with the snap to feature on.

Another addition is the Tools form illustrated below, a companion form to `Wed`.



Tools Form

As its appearance suggests, clicking on this form and then on the target form places a new instance of a control on the target form. It's a quick and intuitive way to place controls. There is a new control as well, a Toolbox, for which the Tools form is itself an example. This control has become popular on "tool bars" to access frequently used procedures. You place a toolbox directly on your form using the Tools form, then specify through its properties, the icons or bit maps to appear. You must compile bitmap and icon resources into the APLGUI.EXE, or use those available. Those appearing on the Tools form are of course readily available, but you will want to create custom icons and bit maps. It is back to your favorite resource editor or the SDK. By the way, the controls now have a sculptured look to them, adding to their professional appearance.

Finally, `Wed` is faster, because it's smarter. When adding or editing a control, `Wed` incrementally compiles the changes into the target form. This is much faster than recompiling the entire form. You will still want to compile the form when removing controls, and to run the form in the test mode from time to time to see how things are working out. In general, I found `Wed` quicker and easier to use.

User Function `Wmake`

Make functions are functions that create a form, from scratch. They begin by deleting all instances of the form, invoke the "New" method, then add each form feature – property and control – to the form. The make function is the ultimate portable representation of a form. It's just another APL function, so you can view and store it like any other function. You can see precisely how the form is created and how every control is endowed, analogous to a batch job. Simply execute the make function and there is the form, ready to go. Sounds neat, yes?

The hard part of a make function is writing the function. Creating the make function is what `Wmake` is for. This utility takes all the pain out of writing make functions. Polish the form and when you're ready, execute

```
□DEF Wmake 'fmMyBestYet'
```

`Wmake` reads the form's definition from the workspace and creates the visual representation of a make function, in this example `fmMyBestYet_Make`. By adhering to the name convention for event handlers, the APLGUI library routines automatically save make functions with the form's event handlers. Try `Wmake` on any APLGUI form, say a demo form. Fire up APLGUI, copy in the APLGDEMO workspace, and pick a form. (For a list of all forms in the workspace, execute `Wforms''`.)

I'm a believer in reading code: your code and code like that from `Wmake`. Reading the make functions for demo forms is an effective tool for learning. In general, reading code is often more efficient for debugging than testing, and it's always enlightening.

Problems can arise when running the make functions created by `Wmake`, so keeping the form in another format as a fallback is a smart idea. After all, the first thing the make function does is destroy the current form. If it fails to properly rebuild the form, the form can be lost. Keep the form in a forms file or save the workspace before executing these make functions. (See `Wsave` and `Wcopy` for managing forms files.) After executing the make function, test the form before burning any bridges.

There is one systematic error that can occur, appearing as an APL stop with the message 'SELECTION ERROR'. I found this error easy to work around. If the stop occurs while the make function is attempting to set the value property for a control, and if that value setting *precedes* the setting of the list property for the same control, rearrange the order of these expressions, clear the suspension, and try again. Apparently without the list property set, any attempt to set a value fails for lack of a structure. Setting the list property first provides such a structure. This problem occurred for a combo box and might occur for other controls as well, especially if your form initializes value properties. Another work around for this problem is to not initialize the value property, but instead set this property in the form's open event handler. This approach lends itself to repeated remaking of the make function without the necessity of editing in a fix.

Another subtle problem can come up with `Wmake` functions and option controls, otherwise known as radio buttons. These controls depend on their tab property to form groups within which at most one can be "on" at any given time. The problem appears when the make function fails to reproduce the original grouping. In my case, the one option group was just disbanded. The `Wmake` function looks correct, but when executed, the tabs don't get assigned as you, and apparently `Wmake`, would expect. I simply adjusted the make function to explicitly set tab properties for the option controls instead of relying on the automatic setting. This cleared up the problem.

WGetOpenFileName, WGetSaveFileName

An earlier column covered these dialogs in detail. There is a small problem plaguing the multiple filename case. If you're planning on using these dialogs, I recommend that you download SWAPL0.ZIP from the Manugistics BBS (301-984-5222, N-8-1) or from the CompuServe codeport forum, library 7. This file contains an expository form discussed in the last **Application Notes** column, and corrected copies of `WGetOpenFileName` and `WGetSaveFileName`. Simply copy these two functions into the WINDOWS workspace and save. These two functions are the only replacement code.

WChooseColor

This support utility is distributed in the WINDOWS workspace and is described on page 1-91 of the Version 5.1 Update Manual; the color selection dialog should look familiar to anyone acquainted with color setting in Windows. This utility offers the user a screen of 48 predefined colors, and the option to define up to 16 additional custom colors. This dialog is another excellent opportunity to use tested code maintained by Microsoft. The calling sequence is straightforward and documented in the Update Manual. In going through this function however, I discovered something counter-intuitive that you might find confusing. Perhaps the following discussion will save you some time.

Windows can represent colors by a 3-vector, (R,G,B), where R = Red, G = Green, and B = Blue intensity, integers in the range 0-255 inclusive. These colors combine as the primary projected light colors to form any visible color, according to 3-color theory that monitors are built upon. For example, (255 0 0) is red, I mean real red, while its compliment, (0 255 255) is cyan.

Windows, and `WChooseColor` in particular, also represents a color as a scalar integer, `C`, defined by the expression:

```
C ← 256 ⊥ B,G,R
```

The order of colors in the above expression is the reverse of what you might expect. According to Manugistic developer Joe Hatfield, the order of byte significance in Intel machines causes this apparent inconsistency. `WChooseColor` returns scalar integer color representations that, when used as color properties, do indeed result in the color selected in the

`WChooseColor` dialog. (The comments in `WChooseColor` even have colors reversed. It really is B,G,R; trust me.)

If you wish to interpret colors in standard (R,G,B) intensity, you may wish to use the following code on the results from `WChooseColor`.

```
C←⌽⍉(3⍴256)⊤C
```

where `C` is a result from `WChooseColor`. `C` becomes a 1x3 or, when the user defines custom colors, a 17x3. Rows correspond to the color selected in the dialog at exit time followed by 16 custom colors (including unused whites); columns correspond to (R,G,B) intensities in conventional fashion. This is of course only necessary if you have need of a color in its (R,G,B) primary components.

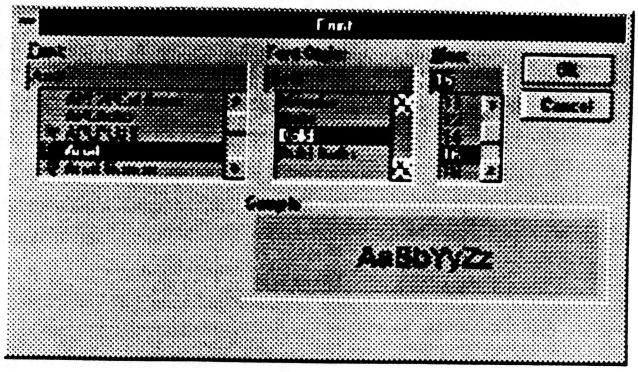
Custom colors are returned from `WChooseColor` as 16 values following the color last selected. If the user defines fewer than 16 new colors, the result is padded by white (16,777,215 = 256⊥ 255 255 255). Custom colors are filled in on the dialog box in a 2x8 array of color squares, in column major order. (See Update Manual Figure 1-6, page 1-91.) In other words, the first column is filled in first, then the second column and so on. `WChooseColor` returns custom colors however in row major order.

Using `WChooseColor` results as right arguments without change will return the custom colors to their original positions. Thus, under normal usage, the order of custom colors is transparent to your application.

If, however, the order in which the user defines custom colors is important to the application, make an appropriate adjustment. For example, if you need to know how many custom colors the user defined, be sure to check all colors returned from `WChooseColor`, not just those preceding the first occurrence of a white. If, say, the user defined four colors in the dialog, those colors would be numbers 1,9,2, and 10 in the 16 vector of custom colors.

WChooseFont

This support function brings up the Windows ChooseFont dialog box routine. It's found in the WINDOWS workspace and documented on page 1-92 of the Update Manual. The dialogue box, shown at the top of the next column, should be familiar to Windows users.



Windows ChooseFont dialog box routine

Font selection is another prime opportunity to take advantage of existing Windows code. Calling this APL function is again straightforward. There are many options for restricting the fonts offered, leading to considerable flexibility. Pick the first nested element from the result for a structure defining the selected font, including size, bold attribute, italics attribute, etc. This structure is compatible with other APLGUI requirements for a font specification. There is no need to list the calling syntax here, as the Update Manual documents this well. It's just important to know that you have easy access to this dialog from the APLGUI.

More to Come

There is more yet in Version 5.1 that will just have to wait. Two big items are the Draw Method for graphics applications, with access to the Windows GDI, the Graphic Device Interface, and the APLWSDK, a tool kit for extending the APLGUI itself. Two additional support utilities, `WPrintDlg` and `Whelp`, round out the basic level of support for Windows applications. We'll cover these functions in later columns.

Using REXX/XEDIT as a Windowing Interface to APL2

Margaret Brooker
Irving, Texas
e-mail: 72124.271@CompuServe.com

If you are using APL2 on VM/CMS then you will already have access to REXX and XEDIT. REXX is IBM's standard language for programming with system commands and XEDIT is a very powerful programmable editor. Together they can be used with APL2 to create a simple screen interface without the need for programming in GDDM or ISPF.

PROFESSIONAL APL PROGRAMMER / ANALYST SEEKS PERMANENT POSITION

Experience:

- 14 years programming and analysis using APL2, VS/APL, and APE (Application Prototype Environment) in several IBM mainframe environments (VM/CMS, VM/XA, and MVS/XA).
- 6+ years in the petroleum industry (Engineering Research and Development) writing applications in the refining simulation and cost estimation fields.
- 7 years experience managing a staff of 10+ people.
- 8 years experience with IBM (and IBM-compatible) personal computers utilizing a wide variety of application software in a Microsoft Windows environment. 6+ years experience using Banyan LAN.
- Excellent verbal and written skills.
- References available upon request.

Contact:

Merritt E. Tilley
10703 North MacArthur Boulevard, # 404
Valley Ranch (Irving), Texas 75063-5226
or telephone (214) 402-8792
or e-mail Internet: 72703.3430@CompuServe.com

This is not a full-fledged screen interface and is not suitable for applications that require a lot of screen input. It is best suited to a simple menu type application where choices or selections are made by moving the cursor about the screen and then pressing PF keys for various tasks. The sample application that we will walk through in this article is a menu to browse all the functions in an APL workspace and then select functions for browsing, by moving the cursor to a function on the screen and pressing a PF key.

There will be two components to the programming: the first will be a set of APL functions stored in a workspace called MENU and the other will be the REXX and XEDIT execs and macros created outside of APL. The first APL function that we will need is one to communicate with CMS from APL under program control. This function has the same effect as the *)HOST* command, but unlike *)HOST* can be invoked from within a function. The function called *CMS* below uses the AP 100 to communicate with CMS.

```
      Z←CMS B;C
[1]   ⍝ Execute a CMS command
[2]    C ← 'CMS'
[3]    → (Z ← 2 ≠ 100 ⎕SVO 'C')/0
[4]    C ← B
[5]    Z ← C
      ∇
```

Also needed are functions to read from and write to flat CMS files. Two functions will be used here for that purpose: *READALL* and *WRITEALL*, using AP 110. Most people will already have their own functions for reading and writing to files, so these will not be listed here. Each function reads from or writes to an entire file which is the actual menu screen.

In this example, we will create a menu of functions within the workspace. The menu screen is composed of a flat CMS file and an XEDIT macro is used to display the menu and allow us to attach different actions to the PF keys. The function *MENU* below does this. It creates the CMS file as a list of functions in the workspace and then writes it to a file called 'MENU FILE A' . The last line XEDITs the file using a macro called MENU.

```
      MENU;R
[1]   R ← CMS 'ERASE MENU FILE A'
[2]   R←'MENU FILE A' WRITEALL ⎕NL 3
[3]   R←CMS'XEDIT MENU FILE A(PROFILE
MENU'
      ∇
```

Later we will write a function *BROWSE_FN*, but for now I will describe the MENU macro. In the macro, we will do some screen formatting using some of the many XEDIT options and apply actions to the PF keys. From CMS (after saving the APL functions) we create the MENU macro from a file called MENU XEDIT. (XEDIT macros must have a filetype of XEDIT.) Within them you can use XEDIT commands and REXX instructions, as well as CMS and CP commands. They are written using REXX programming conventions. The very first line of the macro should be a comment:

```
/* xedit macro to create an APL menu */
```

Now we can start formatting the screen. To remove the prefix area from the XEDIT screen (it is usually set on as a default) :

```
'SET PREFIX OFF'
```

To allow APL characters:

```
'SET APL ON'
```

A scale is usually set on as a default. The following command will remove it:

```
'SET SCALE OFF'
```

The next command will define the position on the screen of the current line in the file being edited as line number 4:

```
'SET CURLINE ON 4'
```

Next we need to determine the number of lines in the screen. This can vary depending on many factors. The XEDIT extract command will retrieve information about the screen into the variable SCREEN. SCREENSIZE is the number of lines in the screen.

```
EXTRACT '/SCREEN'
SCREENSIZE = SUBSTR(SCREEN.1,6,2)
```

Now we can set the message line at the bottom of the screen. This is used for CMS and XEDIT system messages.

```
'SET MSGLINE ON '  SCREENSIZE
```

The very first line on the screen whilst in XEDIT contains information about the file being XEDITed, including file name and size, etc. I find this useful information and don't like to overwrite it. The next line can be set to a heading using the SET RESERVED command. This is a very useful command for screen formatting; it can be used to reserve any line on the

screen and display set text on that line. Here we will use it for a heading and also on the bottom of the screen to show the PF key definitions.

```
'SET RESERVED +2 BLUE REV HIGH Menu of
functions '

'SET RESERVED ' (SCREENSIZE-2) ' BLUE
REV HIGH',

' PF1 = help    PF2 = browse fn   PF3 =
exit to APL   PF12 = exit to CMS '
```

Here we have set the third last line on the screen to display PF key definitions. The second last line is the command line, used for issuing XEDIT commands and the last line has already been reserved for system messages.

Now we need to define the PF keys. Any PF keys that are not set in the macro will revert to the default XEDIT PF key settings. You may need to check what your defaults are. First we set the PF1 key to the help key. When this key is pressed from the MENU macro it will display the file 'HELP FILE A'. This should be a flat CMS file of help text. No XEDIT macro has been specified so the default XEDIT profile will be used.

```
'SET PF1 XEDIT HELP FILE A '
```

The next PF keys will have multiple commands on the same line so that we will need to define a character to use as a command separator. The following command will set the line separator to '#'.

```
'SET LINEND OFF'
```

Later we will write an APL function *BROWSE_FN* which will allow us to browse a function selected from the menu. Here we also use an XEDIT macro called GETPOS to capture the position of the cursor on the screen and then pass it to the APL function. QQ quits XEDIT without saving any changes to file and returns to APL.

```
'SET PF2 GETPOS # QQ # BROWSE_FN'
```

PF3 will exit XEDIT and return the user to APL. Before we leave XEDIT we will erase the file 'MENU FILE A' that we used for our screen.. PF12 does the same actions as PF3 but it also does an *)OFF HOLD* to exit APL. Be careful not to press PF12 if you have made any changes to the workspace!

```
'SET PF3  ERASE MENU FILE A # QQ'

'SET PF12 ERASE MENU FILE A # QQ # )OFF
HOLD'
```

Keys PF7 and PF8 are set to page backwards and forwards in the default profile and so don't need to be set here. The next command closes the definition of the command separator:

```
'SET LINEND ON'
```

This completes the definition of the XEDIT macro MENU. Be sure to save the file as MENU XEDIT A.

The following macro, GETPOS, extracts the cursor position on the screen and pushes it into the program stack. When the APL function *BROWSE_FN* is called, it will pick up this value and use it to determine which row was selected on the screen.

```
   GETPOS XEDIT A

/* getpos macro called by the MENU
macro */
EXTRACT '/CURSOR'
QUEUE CURSOR.3
```

Now we need to write the APL function *BROWSE_FN*, as shown below.

```
      BROWSE_FN;ROW;FN;MENU;R
[1]   ROW ← ⍎ ,⍞
[2]   MENU ← READALL 'MENU FILE A'
[3]   FN ← ⎕CR MENU[ROW;]
[4]   R ← CMS 'ERASE EDIT FILE A'
[5]   R ← 'EDIT FILE A' WRITEALL FN
[6]   R ← CMS 'XEDIT EDIT FILE A(
PROFILE BROWSE'
     ∇
```

Line 1 reads in the row number that was extracted in GETPOS and then pushed into the program stack. Next we read in the file 'MENU FILE A' into a variable *MENU*. The function that was selected is found by using the row number, and a variable containing the text of the function put into the variable *FN*. To browse the function another file is created called ' BROWSE FILE A'. If the file already exists it must be erased before we write out the next one. The last line uses the CMS BROWSE command to view the file, which contains the code of the function that has been selected from the main menu. This XEDIT macro is shown below.

```
   BROWSE XEDIT A
/* macro to browse a function */
'SET  LINEND  OFF'
'SET  PF3  QQ# MENU'
'SET  LINEND  ON #'
```

Now the MENU application is ready to be run - simply `)LOAD MENU` from APL, type in `MENU` and you will be presented with a screen displaying all the functions in the workspace. By moving the cursor to a function name on the screen and pressing PF2, you can display the contents of the function that was selected. If there is more than one page of functions in the workspace, you can page through them with keys PF7 and PF8.

There are many possible extensions that can be easily made to the application. You may want to exclude certain functions from being displayed, for example the functions `MENU`, `BROWSE_FN` and `CMS`, which can be done by excluding them from the function list that is created in MENU. Another useful addition would be the ability to edit a function from the menu. This could be achieved by changing the function `BROWSE_FN` to `□CR` the function when exiting from the BROWSE macro. Other possibilities include adding more information about the functions to the screen, such as the date the function was last created, etc. You could also add the ability to print functions: by moving the cursor to a function on the main menu screen and pressing another PF key, a file could be created with the function code, and the file sent to a default printer. It is also possible to invoke the application directly from CMS, by creating a REXX EXEC to pass the workspace name as an argument to the APL2 command.

I hope this article has presented to you some of the many interesting possibilities that are available using APL2, REXX and XEDIT.

THE SOUNDEX METHOD

Todd B. Marshall
Conroe, Tx.
e-mail: 70544.715@CompuServe.com

Whether you are dealing with an Airline Reservation System or a Library of Congress research project, there is often a need to search text information. Misspellings and spelling variations make character by character matching strategies inappropriate.

You may be searching for a passage like "counterfeit pseudo-matrimonial ties" or you may be searching for a name like "Smyth". There is a chance that your representation of words will differ from the text. You may be an excellent speller but the document you search may be written by someone less skilled. In any event, a strategy which matches sequences of characters can leave you disappointed.

To address problems of this type, in 1918, Margaret Odell and Robert Russel patented a method for encoding surnames called the "Soundex" method. It is a four step process:

1. Retain the first letter of the name, and drop all occurrences of a,e,h,i o,u,w,y in other positions.

2. Assign the following numbers to the remaining letters after the first (1: b,f,p,v) (2: c,g,j,k,q,s,x,z) (3: d,t) (4: l) (5: m,n) (6: r)

3. If two or more letters with the same code were adjacent in the original name (before step 1), omit all but the first.

4. Convert to the form "letter, digit, digit, digit" by adding trailing zeros (if there are less than three digits), or by dropping rightmost digits (if there are more than three).

For example, the names Euler, Gauss, Hilbert, Knuth, Lloyd, and Lukasiewicz have the respective codes E460, G200, H416, K530, L300, L222. While these are the same codes generated by Ellery, Ghosh, Heilbronn, Kant, Ladd, and Lissajous, by and large the Soundex code greatly increases the chance of finding a name in one of its disguises.

The APL function `SOUNDEX` at the end of this article yields Soundex codes given words as a right argument. There are some difficulties with this function.

It has no provision for non-alphabetic characters.

It cannot process a string of words yielding a string of codes.

After working with Soundex, I found that the invention itself is wanting. There are character combinations in the English language that bear no resemblance to their sounds. For example, the "ph" in phantom and graph; and the "ps" in psychology and pseudonym. I would want to change "ph" to "f" and "ps" to"s".

Further, suffixes are often added to the words that trip me up in my search. For example, Fox gives a code of F200. Foxes gives a code of F220. Foxed gives a code F230. I drop suffixes of "es", "er", and "ed". In fact, dealing with plurals in general leads me to drop the trailing "s" off all words.

Often I am interested in root words. Adding "ly", "able", "ing" or "ion" for proper usage in context makes the root words difficult to find in a search. I have experimented with dropping these suffixes with generally favorable results.

Embedded apostrophes have the effect of splitting words. For example, the word "weren't" becomes "weren" and "t" when replacing non-alphanumerics with blanks. So I remove apostrophies rather than blanking them.

I also find that limiting the code to four characters limits the usefulness of the method for longer words. I find it suggesting absurd matches. So I like a code of more than 4 character resolution.

Finally, I often find I want to search for numbers as well. For example, I may want to find documents containing 386SX or 486DX.

The 4-character Soundex code requires 32 bits to store. But if you consider there are 36 first letters (alphabetics plus numerics) and only 16 sound codings (including numerics) in Soundex you see that you can effectively store up to 7 characters in a 32 bit integer.

Considering this and through "software sculpturing", I produced a function *BULK_SOUNDEX* that is serving me well. It is presented at the end of this article. The function takes a character string as its right argument and returns an integer vector of sound codes for the string.

Typically the argument is a paragraph of a document. Filing documents by paragraph gives me a good balance of object size against index size.

This *BULK_SOUNDEX* function is quite fast at producing all codes for a paragraph. It is comparatively slow at producing a code for a word or short sequence of words. I am willing to trade off a little speed when interacting with the user for bulk speed coding documents. Actually, the speed penalty is imperceptible to the user. But the improvement is as much as a factor of 10 over other algorithms when dealing with bulk data.

In using the function, you see that it has a compression effect. All the unique codes produced by lengthy documents usually number less than 2,000. If you set up a database keyed by sound code with documents or paragraphs linked to this code, you can quickly find passages containing the codes. The technique requires minimal indexing storage.

For example, the word "phenomenon" produces a code of 214076844. A two-thousand element integer vector containing all codes in all documents would require just 64K of storage. Using dyadic iota, the position of 214076844 in such a vector could reference a component in a file. This component could contain locations of all documents (paragraphs) containing this code. Thus I could find all the documents containing the word "phenomenon" through a simple dyadic iota and two component reads.

The function I produced uses four ancillary functions:

DEB: An assembler function that removes extraneous blanks (consecutive blanks are not permitted)

TRANSLATE: An assembler function that translates characters within a character vector

TEXTREPL: An assembler function that replaces character strings within a character vector

WHERE: An assembler function that returns indices of 1's in a bit vector

These functions can be found in Manugistics' ASMFNS workspace.

Committing the *BULK_SOUNDEX* algorithm to machine language would yield blinding speed and a fraction of the code. That is my next task.

```
    ∇ R←SOUNDEX W;A;G;F;I
[1]   A←'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
[2]   G←52⍴'01230120022455012623010202'
[3]   F←W[1] ⍝ Retain the first letter
[4]   I←A⍳W ⍝ Get the index into the coding vector
[5]   R←G[I] ⍝ Convert letters to codes
[6]   R[1]←'7' ⍝ Set delimiter for duplicate test
[7]   R←1↓(R≠1⌽R)/R ⍝ Remove duplicates
[8]   R←R~'0' ⍝ Remove vowels and other discards
[9]   R←4↑F,R,'000' ⍝ Assemble Soundex Code for the word
    ∇
```

```
    ∇ R←BULK_SOUNDEX S;IDX;FL;RT;F;⎕IO
[1]   ⍝ SOUNDEX ALGORITHM ON STRING
[2]   ⍝ Returns vector of codes R for words in character vector S
[3]
[4]   ⎕IO←1◊R←0 ◊S←' ',S,' ' ⍝ Assure leading and trailing blanks
[5]      ⍝ Translation of letters to uppercase
[6]   TR←256⍴' '◊TR[¯1+,(⎕AV⍳'Aa')∘.+⍳26]←⎕AV[52⍴¯1+(⎕AV⍳'A')+⍳26]
[7]      ⍝No translation of digits.  ⎕AV[40] is word delimiter
[8]   TR[40]←⎕AV[40] ◊TR[(¯1+⎕AV⍳'0')+⍳10]←⎕AV[¯1+(⎕AV⍳'0')+⍳10]
[9]   S←TR TRANSLATE S
[10]     ⍝ Translate Sound, suffix, and quote
[11]  TR←'\ PS\ S\PH\F\S \ \ED \ \ER \ \LY \ \ABLE \ \ING \ \ION \ \''\'
[12]  S←TR TEXTREPL S
[13]     ⍝ Remove extraneous blanks
[14]  S←' ',DEB S◊→(1=⍴S)⍴0
[15]     ⍝ Word first character translation to domain 2...37
[16]  F←'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'⍳S[IDX←1+WHERE S=' ']
[17]  WORDS←(S=' ')⌿⎕PENCLOSE S
[18]     ⍝ Translate to sound codes
[19]  TR←256⍴⎕AV[1] ◊TR[⎕AV⍳'BFPV']←⎕AV[2]
[20]  TR[⎕AV⍳'CGJKQSXZ']←⎕AV[3]◊TR[⎕AV⍳'DTLMNR']←⎕AV[4 4 5 6 6 7]
[21]  TR[⎕AV⍳'0123456789']←⎕AV[7+⍳10]
[22]  S←TR TRANSLATE S
[23]     ⍝ Replace first character
[24]  S[IDX]←⎕AV[40]
[25]     ⍝ Remove contiguous duplicates
[26]  S←((S≠1⌽S)∨(S=⎕AV[40]))⌿S
[27]     ⍝ Remove untranslated characters (including vowels)
[28]  S←S~⎕AV[1]
[29]     ⍝ Convert each word to soundex integer
[30]  S←1↓¨(S=⎕AV[40])⌿⎕PENCLOSE S
[31]  R←18 ⊥¨6↑¨¯1+¨(⊂⎕AV)⍳¨S
[32]  R←R+F×18*6
    ∇
```

⍝ *EXTENSION NOTES*

by Gregg Taylor

TOPIC: Don't overlook these shortcuts!

APL: concise; English: well, it can go on and on, around and about, and eventually get across a message; and most other languages (save COBOL) fall somewhere in between. Sometimes I think English speaking people who write APL feel uncomfortable unless they can be loquacious. At least this seems to be the case in a lot of code that I see. Of course, it is really a matter of whether the code-writer has seen an easier way to write certain code snippets. This article offers an opportunity to see some better ways.

Before starting, I would like to point out that the following are not necessarily idioms of APL; rather they are shorter ways to write certain code strings. There are undoubtedly variations on the strings I will be using, but I haven't seen it all, and what follows is fresh on my mind. I have tried to split them up into logical categories to help draw out similarities.

Alternatives to *reshape* (⍴):

Creating a column vector in APL2:

```
COLVEC←((ρVEC),1)ρVEC                    ⍝ ONE WAY
COLVEC←,['']VEC                          ⍝ NEW WAY
```

Making strings of repeating numbers:

```
VEC←(4ρ1),(2ρ0),(6ρ2),(4ρ0),5ρ4          ⍝ START WITH THIS
VEC←4 2 6 4 5/1 0 2 0 4                  ⍝ MOVE TO THIS, OR...
VEC←4 ¯2 6 ¯4 5/1 2 4                    ⍝ IN APL2
```

Reducing dimensionality in APL2:

```
MAT←(↑ρARRAY3D)(×/1↓ρARRAY3D)ρARRAY3D
MAT←,[⎕IO+1 2]ARRAY3D
```

Using scalar extension:

```
MAT[VEC1;VEC2]←((ρVEC1),ρVEC2)ρ'*'       ⍝ INSERTING DECORATIONS
MAT[VEC1;VEC2]←'*'

MAT←MAT,[⎕IO](1,1↓ρMAT)ρ'-'              ⍝ ADDING A ROW
MAT←MAT,[⎕IO]'-'
```

This branch is a little shorter:

```
→(0=ρ,B)ρL1                              ⍝ BRANCH AROUND NULL CASE
→(ρ,B)↓L1                                ⍝ TEST NOT NEEDED
```

Simplify *catenation* (,):

Laminate isn't that hard:

```
MAT←(((ρVEC1),1)ρVEC1),1                 ⍝ ADD SECOND COLUMN OF 1S
MAT←VEC1,[⎕IO+.5]1
```

Keep dimensionality, rather than add it:

```
MAT2←MAT1[;2],[⎕IO+.5]VEC                ⍝ CREATE 2-COLUMN MATRIX
MAT2←MAT1[;,2],VEC
```

Add dimensionality early, rather than late:

```
MAT2←(+/MAT1[;⍳3]),[⎕IO+.5] +/MAT1[;3+⍳3]
MAT2←(+/MAT1[;1 3ρ⍳3]),+/MAT1[;3+⍳3]
```

In APL2, use the new axis capabilities:

Don't worry about the number of columns:

```
MAT←((3+↑ρMAT),1↓ρMAT)↑MAT               ⍝ ADDING 3 ROWS
MAT←(3+↑ρMAT)↑[⎕IO]MAT
```

This is a jewel we don't need to see any more:

```
MAT←MAT×⍉(⌽ρMAT)ρ.05 .06 .07             ⍝ EACH ROW TIMES A NUMBER
MAT←MAT×[⎕IO].05 .06 .07                 ⍝ THAT'S A RELIEF
```

Combine adjacent dimensions (see above for similar usage):

```
MAT←(×/2↑⍴ARRAY4D)(×/¯2↑⍴ARRAY4D)⍴ARRAY4D
MAT←,[⍳2],[¯2↑⍳4]ARRAY4D
```

***Selective assignment* (←) really makes some code easier:**

Replace one character with another in a matrix:

```
TVEC←,MAT                              ⍝ THESE 3
TVEC[(TVEC='¯')/⍳⍴TVEC]←'-'            ⍝ LINES NOW
MAT←(⍴MAT)⍴TVEC                        ⍝ BECOME 1:

(('¯'=,MAT)/,MAT)←'-'
```

Don't worry if it doesn't fit:

```
VEC[3↓⍳6⌊⍴VEC]←(0⌈¯3+6⌊⍴VEC)↑100 110 120
(3↑3↓VEC)←100 110 120
```

Dyalog APL implements *assignment* (←) as an operator:

Increment a counter:

```
CNTR←CNTR+1                    ⍝ A WAY
CNTR+←1                        ⍝ THEIR WAY
```

Add a row to a matrix:

```
MAT←MAT,[⎕IO]'='
MAT⍪←'='
```

Apply a factor to a sub-array:

```
MAT[VEC1;VEC2]←MAT[VEC1;VEC2]×.01
MAT[VEC1;VEC2]×←.01
```

Some miscellaneous uses of primitives:

Without (~) isn't that new (but is not in Iverson APL):

```
NEWVEC←(~VEC1∈VEC2)/VEC1               ⍝ IF IN VEC2, REMOVE IT FROM VEC1
NEWVEC←VEC1~VEC2
```

Element (∈) can do lots of things:

```
INDICES←(~+/[⎕IO]VEC1∘.=VEC2)/⍳⍴VEC2
INDICES←(~VEC2∈VEC1)/⍳⍴VEC2

→(1≤+/BOOL)↑L3                         ⍝ ANY 1S IN HERE?
→(1∈BOOL)↑L3
→BOOL/L3                               ⍝ IF 'BOOL' NOT TOO LONG
```

As you may guess, this could go on and on, like a bad English novel. The point is that APL is concise, is becoming more concise, and should be used that way. We all have our round-about ways of writing; these give us our style and may even improve readability. But it just may be worth your while to think a bit about even your small code segments, and how they can be made simpler.

KISS
SOFTWARE INTERNATIONAL

presents

OOF: OBJECT ORIENTED FILING
for MANUGISTICS' *APL*PLUS II/386 V5*

43 CASPER COURT
CONROE, TX. 77302
(409)273-2100

$14.95

12 OBJECT FILE FUNCTIONS
38 DERIVED OBJECT FUNCTIONS
14 APPLICATION MANAGEMENT FUNCTIONS
22 SHARE FILE FUNCTIONS
14 NATIVE FILE FUNCTIONS
11 ANCILLARY FUNCTIONS

90 PAGE REFERENCE AND TUTORIAL

OBJECT ORIENTED FILING FUNCTIONS

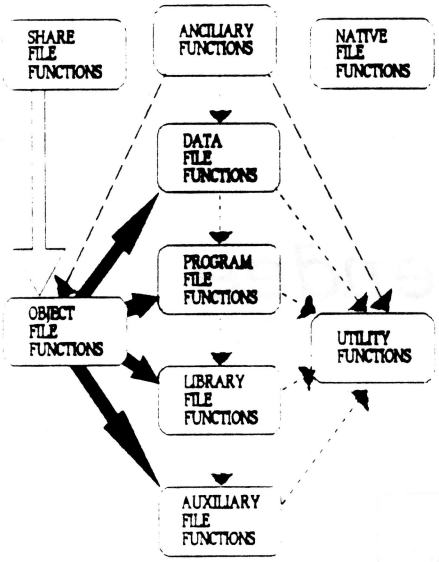
BASE \ DERIVED	*AUXILIARY*	*PROGRAM*	*LIBRARY*	*DATA*
OOFCI	AUXCI	PGMCI	LIBCI	DATACI
OOFCN	AUXCN	PGMCN	LIBCN	DATACN
OOFCREATE				
OOFFILE	AUXFILE	PGMFILE	LIBFILE	DATAFILE
OOFGET	AUXGET			DATAGET
OOFNAMES	AUXNAMES	PGMNAMES	LIBNAMES	DATANAMES
OOFPGMGET	AUXPGMGET	PGMGET	LIBGET	
OOFPGMGETL	AUXPGMGETL	PGMGETL	LIBGETL	
OOFPGMPUT	AUXPGMPUT	PGMPUT	LIBPUT	
OOFPUT	AUXPUT			DATAPUT
OOFRMV	AUXRMV	PGMRMV	LIBRMV	DATARMV
OOFTS	AUXTS	PGMTS	LIBTS	DATATS
		PGMRUN		

Other Functions

Utility Functions	*Ancillary Functions*
FILECHGS	CURRDIR
FILECMPR	DAB
FILELIST	DLB
FILEPNCPR	DTB
FILEGLOBALS	FL2NA
Program Management Functions	FL2PPN
PGMDOC	KISSAKI
PGMFNCPR	L2NA
PGMGLOBALS	NAGRADE
PGMLOCK	NASORT
PGMSEARCH	RMVDUP
PGMSMRY	
PGMUDWS	
PGMUNLOCK	
PGMXREF	

PRIMITIVE WRAPPER FUNCTIONS

Sharefile	*Native*
FACCESS	
FAPPEND	NAPPEND
FDUMP	
FCLEAR	NCLEAR
FCOMPRESS	
FCREATE	NCREATE
FDROP	
FDUMP	
FDUPFROM	
FERASE	NERASE
FILES	NFILES
FINIT	NINIT
FNAMES	NNAMES
FRDCI	
FREAD	NREAD
FRENAMETO	NRENAMETO
FREPLACE	NREPLACE
FRESIZE	
FSIZE	NSIZE
FSTIE	NSWAP
FSWAP	NTIE
FUNTIE	NUNTIE



YES! Please send me _____ copies of OOF

☐ **Please bill me for $14.95 plus $3.00 postage and handling charges.(USA Only)**

☐ **Check or money order enclosed with this order form**
(We pay shipping on prepaid domestic orders!)

Name: _____

Company: _____

Street: _____

City: _____ **State:** _____ **Zip:** _____

Country: _____

Phone: (___) - _____

To validate your order please sign here: _____

Mail or call: KISS Software International
43 Casper Court
Conroe, Texas 77302
Phone: (409)273-2100

☐ *3-1/2"*

☐ *5-1/4"*

I understand that if I am not fully satisfied, I may return the software within 15 days, without obligation, and KISS will cancel my bill immediately.

SouthWest APL Users' Group
Membership Application

Name: _____

Company: _____

Address: _____

_____

_____

Telephone: _____ (home) _____ (work)

E-mail: _____

Membership fees, per year (includes subscription to *the APL perspective*):

U.S.A. .....$12.
Non - U.S.A. postal supplement .....$ 6.
Extra Donation.....____

Please mail this application and the applicable dues to: Stuart Yarus
(Checks should be made payable to "SWAPL")
P. O. Box 210367
Bedford, Texas 76095
U. S. A.

Stuart Yarus, Editor
the APL perspective
P. O. Box 210367
Bedford, TX 76095
U. S. A.

*** *Address Correction Requested* ***

APL93 Attendee

A red dot next to your name indicates that this is your LAST newsletter. Please renew your membership today!